Communications of
HUAWEI RESEARCH

December 2022
Issue 3

HUAWEI

SOFTWARE IS EATING THE WORLD

AND

OPEN SOURCE IS EATING SOFTWARE

E-mail: HWResearch@huawei.com

Editorial Note

This is the "Software & Theoretical Computer Science" issue of *Communications of HUAWEI RESEARCH*. Software is the core infrastructure of the digital world and is changing the digital world and physical world. To embrace the digital world, many modern enterprises and even countries are making relentless efforts in building basic software capabilities, in pursuit of breakthroughs in core software technologies.

In contrast to physical objects, software is a logical entity that is invisible and intangible. It is also probably the most complicated system designed by humans, as large software systems can have millions of lines of code. The development of software follows its unique rule. There is no silver bullet that solves all problems. Software includes basic software, application software, and utility software, and is oriented to smart devices, embedded devices, cloud, enterprise IT, and many other industries. Software development for different industries varies, and the ecosystem and requirements are also different. The life cycle and supply chain management of software is unique. To develop good software, we need to study and apply the continuously evolving theories and processes of software development.

Huawei is continuously investing in software and related theoretical research. Huawei has built its competence in various industries, including device, connection, computing, cloud, and intelligent driving, and is continuously improving software engineering and trustworthiness capabilities through trustworthiness and software engineering capability transformation. In addition, Huawei has established two ecosystems based on EulerOS and HarmonyOS, which facilitate the construction of open source communities such as openEuler, openGauss, MindSpore and OpenHarmony.

This issue of *Communications of HUAWEI RESEARCH* summarizes Huawei experts' opinions and latest achievements in basic software, software theories, and open source community construction.

In the basic software part, *Operating System Evolution: History, Status Quo, and Prospect* looks into the development, opportunities, and challenges of operating systems from the perspectives of industry evolution and hardware evolution, while also introducing the practices of openEuler and OpenHarmony in light of the technology evolution. GaussDB Kernel is a distributed database platform for enterprises developed by Gauss Dept of Central Software Research Institute. The paper *GaussDB: Cloud-Native Distributed Database* describes the architecture and key features of GaussDB Kernel. *Open Source Ecosystem of AI Systems: Experience on MindSpore* introduces the architecture and core technologies of the deep learning framework MindSpore, and elaborates on the practices in the ecosystem of AI systems. Following the summary and analysis of the technology evolution of compilers, *Huawei BiSheng Compiler Innovations and Practices* illustrates the technical innovation and practices of Huawei BiSheng compiler, analyzes its core features, and proposes the direction for evolution and innovation.

In the theoretical computer science part, *Recent Advances in Online Matching* takes you through the latest achievements in online matching. *Learning Augmented Algorithm Design for Combinatorial Optimization* presents the research progress in using AI technologies to solve combinatorial optimization problems. Solvers are a very important type of utility software, and lay the foundation for industry software. *Combining Random Walk with a Fitness Function for Boolean Satisability* introduces the breakthrough of applying random walk in SAT solvers. Domain Theory sets the mathematical foundation for programming languages and their analyzers. *Domain Theory Meets Interaction* reviews the development of Domain Theory from 1960s and explains how it is used to describe interaction and concurrency with the evolution of programming models.

"Software is eating the world, and open source is eating software." *Formulation of Open Source Strategies and Measurement of Community Building* shows how enterprises can take advantage of open source opportunities to develop the ecosystem economy and promote industry digital transformation from the aspects of open source strategy formulation and open source community construction of enterprises.

There are many excellent papers in this issue. They focus on key challenges in the software industry, and are in line with industry trends. We hope this issue will make for an inspiring read. Your comments and suggestions are most welcome.

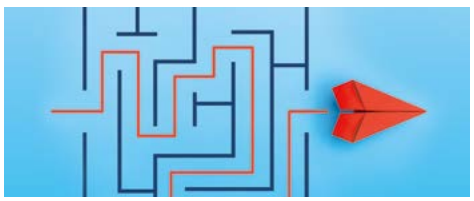
Haibo Chen
Huawei's Chief Scientist for Basic Software

Pinyan Lu
Huawei's Chief Scientist for Theoretical Computer Science

Basic Software

CONTENTS

Theoretical Computer Science



Software Technology and Ecosystem

Operating System Evolution: History, Status Quo, and Prospect

Haibo Chen [1], Zhiyang Qian [1], Ning Jia [1], Xinwei Hu [1], Yi Li [2]
[1] Central Software Institute
[2] Consumer BG Software Engineering Dept

Abstract

Operating systems (OSs) serve upper-layer applications, manage and expose lower-layer hardware capabilities, and enable hardware and application ecosystems. This paper analyzes the development, innovation opportunities, and technical challenges of OSs from two dimensions: industry application evolution and hardware evolution. This paper also elaborates on some practices of openEuler and OpenHarmony.

Keywords

OS, openEuler, OpenHarmony

1 Introduction

An operating system (OS) is defined as "a kind of system software that manages hardware resources, controls program running, improves the man-machine interface, and provides support for application software" in *Encyclopedia of Computer Science and Technology (Third Edition)* [1]. Since the first OS was developed, both the connotation and denotation of OSs have been continuously expanding. An early OS contained only the OS kernel and original Shell (e.g., command line terminal). In comparison, a modern OS supports more functions, such as OS services that extend OS kernel management and abstract hardware resources, and an application framework that creates an execution environment for applications and manages application execution, as shown in Figure 1.

An OS acts as an intermediary between applications and hardware. It provides runtime and development environment services for applications, and serves as the portal for application ecosystems and cloud services in certain scenarios. In addition, an OS efficiently and securely manages hardware, maximizes the potential of hardware, and enables hardware ecosystems.

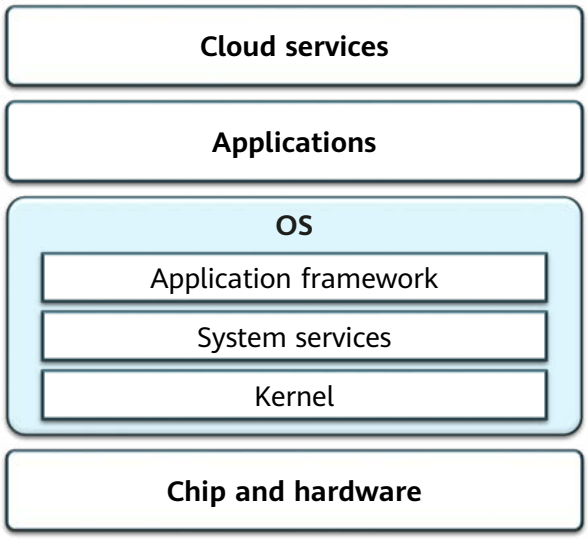


Figure 1 Positioning of the OS in the entire computing system

In this paper, we will start by reviewing the development of OSs from two dimensions: industry application evolution and hardware evolution. Then, we will discuss the prospect of OS innovations driven by application scenarios and that driven by hardware. Finally, we will introduce some innovative practices of openEuler and OpenHarmony.

2 OSs in Industry Evolution

2.1 Birth and Development of OSs with Industry Waves

Historically, notable OSs were born and developed along with industry waves and promoted industry advancement, as shown in Figure 2.

In early computers, software and hardware were deeply coupled, resulting in highly specialized and inefficient application program development. The requirements for facilitating computer use and computer-based application development directly promoted the birth of programming languages and OSs, and the continuous decoupling of software and hardware.

The GM-NAA I/O system [2] developed in 1956 was considered the prototype of modern OSs. It was essentially an input and output management system. It was not until the 1960s that a real OS was developed. Before that, the software and hardware of a mainframe were deeply coupled. It took years to develop and launch software that matched already-developed hardware, slowing down

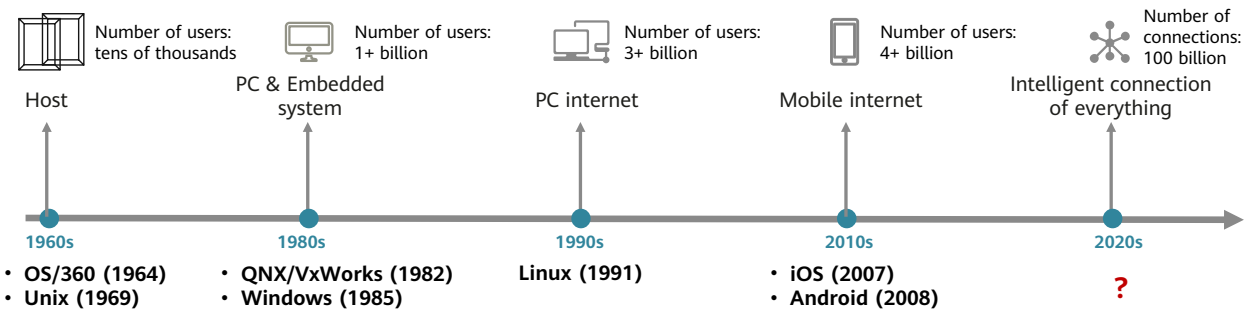


Figure 2 Symbiotic development history of OSs and industry applications

the go-to-market pace of the entire system. OS/360 [3], announced by IBM in 1964, was the first OS to decouple software from hardware. The complex R&D process of this OS inspired the creation of the pioneering book *The Mythical Man-Month: Essays on Software Engineering* [4], which is still relevant even today in helping us understand the complexity involved in software. In 1969, Ken Thompson and Dennis Ritchie at Bell Labs, considering the Multiplexed Information and Computing Service (Multics) [5] OS they were working on too complex, simplified its design concept and developed the Unix OS (originally called Unics, where Uniplexed replaced Multiplexed in Multics) [6]. Based on the concept of Unix, a series of OSs, including Linux, were developed subsequently.

The OSs launched in the 1980s promoted the evolution from midrange computers to PCs and embedded systems. For example, QNX [7], which was developed in 1982, has been widely applied in embedded systems, and is currently used extensively in the intelligent vehicle field. VxWorks [8], also developed in 1982, is popular in embedded systems and industrial scenarios. Since its birth more than three decades ago, Windows has promoted the development of PCs and dominated the markets of PCs and PC servers. In addition, the Linux system — developed in 1991 — has driven the development of PC servers and cloud computing.

The emergence of mobile OSs (e.g., iOS and Android) has promoted mobile devices such as cell phones to evolve from feature phones to smart phones. iOS and Android have become the de facto OSs in the mobile internet era.

In the era of intelligent connection of everything, with the gradual convergence of information technology (IT), communication technology (CT), and operational technology (OT), and the development of technologies such as artificial intelligence (AI) and 5G, OSs will face new opportunities and challenges.

2.2 Success Factors of OSs Have Kept Evolving with the Industry

The value and success factors of OSs have continued to evolve. The positioning of OSs has gone through the following four phases: as an accessory of hardware, as an independent software product, as a portal for ecosystems and cloud services, and as an intelligence enabler for various industries (see Figure 3).

Early OSs (IBM OS/360, early Unix, etc.) served as an accessory of hardware, and therefore their value depended on the value of hardware. For example, IBM OS/360 exhibited its value as an accessory of IBM mainframes. At that time, the performance of hardware devices was severely constrained, making performance the key success factor of OSs in this phase.

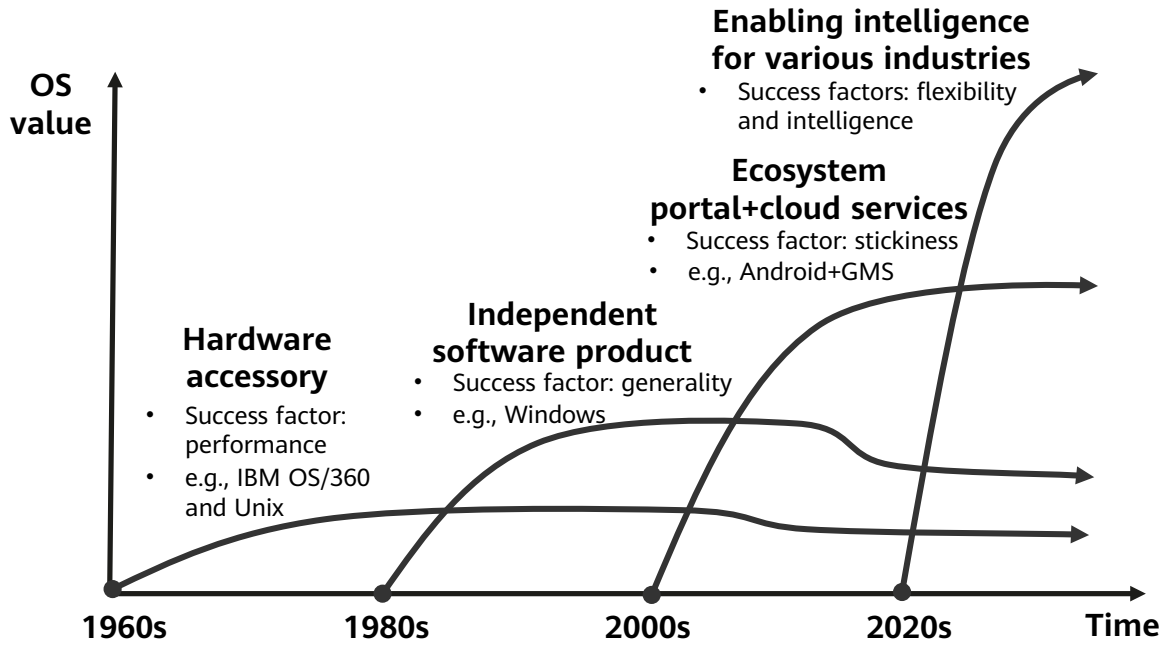


Figure 3 Evolution of the value and success factors of OSs

Since the 1980s, OSs have gradually become an independent software product. Windows is a typical example in this phase. Software providers can gain significant value by selling software licenses, because software is highly reproducible. This is why Microsoft has been one of the world's most valuable IT companies for so long. In this phase, generality was the key success factor of OSs. The compatibility of Windows with different types of devices and applications promoted the development of OSs in the PC era.

Since 2000, the business model of OSs has gradually evolved to provide a portal for ecosystems and cloud services. Apple's iOS, iCloud, and App Store combination and Google's Android, GMS, and Google Play combination are two typical examples. An ecosystem portal and cloud connection form a powerful ecosystem. The main success factor is ecosystem stickiness.

In the future, digital will be brought to every person, home, and organization for a fully connected, intelligent world. The key to achieving this vision is to enable intelligence for various industries — the core success factors of OSs are flexibility and intelligence.

Note that the preceding four modes that emerged in the four phases are not substitutes for each other. Instead, they will all continue to evolve, and the early phases will remain in place for a long time. In the future, we will see the four modes coexist. For example, OSs will continue to support the competitiveness of device products. The pattern in which OSs function as the portal for ecosystems and cloud services is thriving. In addition, the industry is actively exploring how to enable intelligence for various industries.

Device, IT, and CT technologies have gradually been converged to enable intelligence for various industries. To some extent, OSs will take on the new missions of human-machine-material convergence and cloud-network-edge-device synergy in the future, as shown in Figure 4. Faced with static, isolated network domains with secure physical resources in closed scenarios, OSs are static, closed, and restricted systems. In such scenarios, small-scale sampling data analysis is applicable. However, in scenarios oriented to various industries, OSs need to evolve toward being dynamic, open, and scalable. In such scenarios, full-sample large-scale data analysis is applicable, and the security of virtual resources needs more attention [9].

3 Prospect of Business Scenario-driven OS Innovations

3.1 Paradox of Classis Insecta Calls for OSKA Architecture Innovations

Intelligent connection of everything brings numerous changes. One notable change is that products and

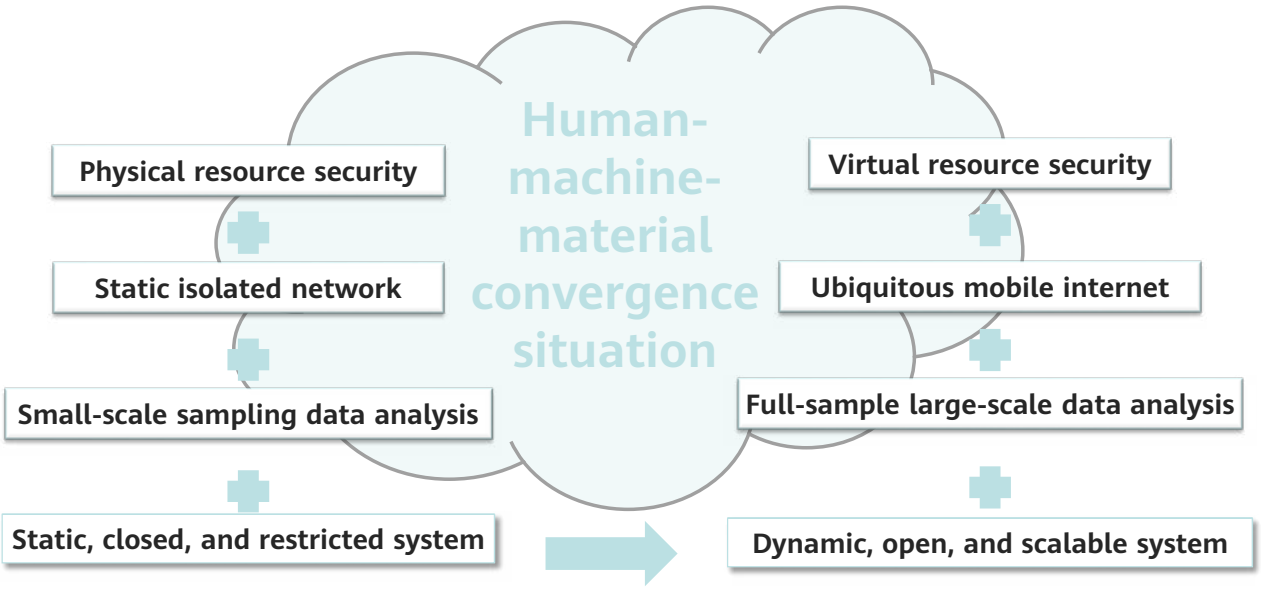


Figure 4 New missions of human-machine-material convergence and cloud-network-edge-device synergy [9]

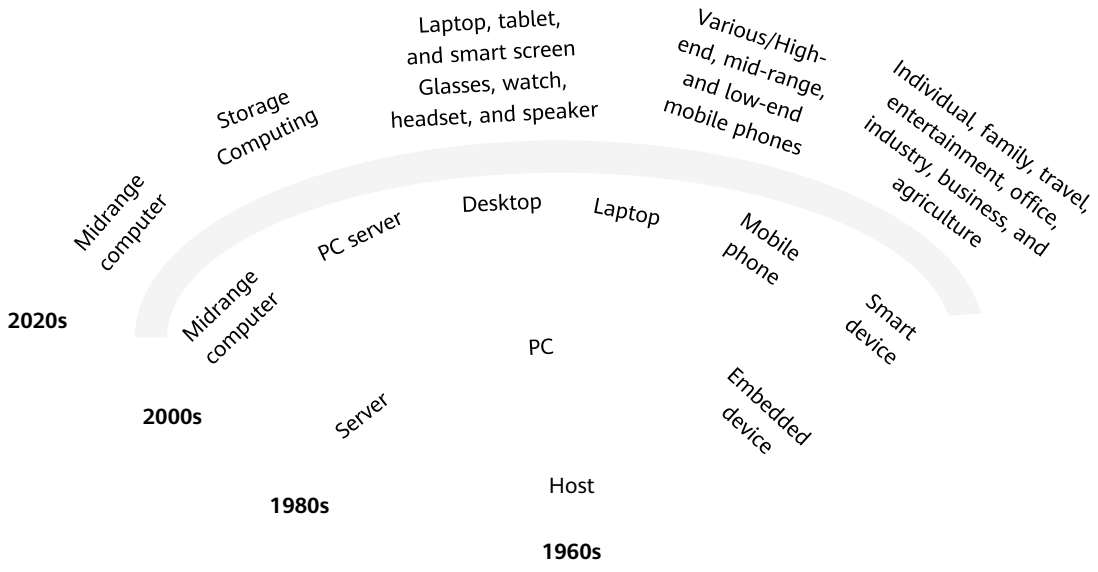


Figure 5 Explosive evolution and hybridization of hardware forms

application scenarios will evolve and hybridize explosively. When a traditional device is hybridized and integrated with other devices, new hardware forms that have higher added value than the original hardware may emerge, as shown in Figure 5.

Paradox of Classis Insecta is a vivid metaphor put forward by Ken Sakamura, a professor at Tokyo University, to describe the contradiction between individuality and generality. The more than 5000 species of mammals on Earth can be compared to traditional devices such as servers, PCs, and mobile phones, while the more than 1 million species of insects can be compared to a variety of devices in the era of intelligent connection of everything. On the one hand, the cumulative value brought by a larger number of devices for intelligent connection of everything may exceed that brought by PCs and mobile phones. On the other hand, costs cannot be significantly cut due to the lack of hardware, software, and applications that can be replicated in large quantities, making it difficult to expand the market.

The continuous innovation of hardware forms in the era of intelligent connection of everything brings the Paradox of Classis Insecta challenge to OSs. Addressing the differences in resource size, function, performance, security, and ecosystem across a plethora of different scenarios is therefore an urgent requirement. OSs must implement component-based decoupling and on-demand combination to ensure architecture consistency and facilitate unified maintenance. Flexible combination can also be used to resolve a number of individual problems. We refer to the OS architecture that meets such design intent as the "One OS Kit for All (OSKA) architecture".

The goal of the OSKA architecture is OSKA. Specifically, we intend to build an OS capability set (OS Kit) that can be flexibly assembled, instead of a rigid one. Based on a highly elastic architecture, the capabilities in the OS Kit are combined into scenario-specific OSs that meet the requirements of various industry scenarios, thereby alleviating the Paradox of Classis Insecta challenge.

3.2 New Computing Forms Call for Innovative Computing Abstraction of OSs

Figure 6 shows the history of two mainstream computing abstractions: virtualization and containerization.

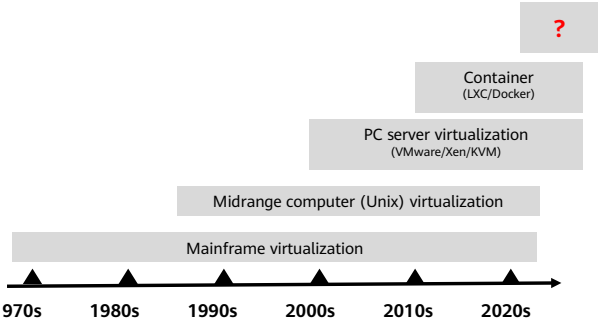


Figure 6 Evolution of runtime abstraction in OSs

Virtualization emerged in the 1970s. While IBM OS/370 was the first OS to provide virtualization capabilities, VMware — founded in 1998 — ushered in the era of PC server virtualization. Containerization began to take shape in the 1970s. For example, Unix V7, which was released in 1979, provided an early isolation environment. Following that, the industry-leading Linux container (LXC) and Docker emerged in 2008 and 2013, respectively. To some extent, both virtualization and containerization enabled the cloud computing revolution.

However, with the rise of function computing and edge computing, the limitations of computing abstractions such as virtual machines (VMs) and containers have become more noticeable. Take function computing as an example. New requirements for the runtime environment are becoming increasingly prominent, including but not limited to the following:

- 75% of function computing has a maximum execution time of only 10 seconds, and the runtime environment is expected to be extremely lightweight and have an ultra-short life cycle.
- 81% of function computing features low-frequency invoking (less than once per minute on average), and requires the resident memory to be reduced more efficiently to improve resource utilization.
- Traditional branch prediction has a high error rate when dealing with short functions, and therefore a more effective new scheduling mechanism is required.
- More open scenarios pose higher requirements for security isolation [10].

These new requirements for OSs differ greatly from the requirements for traditional OSs. The VM- and container-based abstraction features slow startup and increase runtime system overhead, and the security isolation capability needs to be improved. These requirements of new computing forms demand new runtime abstraction in the OS field.

3.3 Contradiction Between System Complexity and Agility Has Promoted the Emergence of Learned Systems

The contradiction between system complexity and agility has spawned the learned system, which is a combination of AI technologies and systems. A closed system uses a stable, closed OS with definite results. In such an OS, the whole is equal to the sum of its parts, and therefore experience-based optimization is applicable. However, for an open, growing, and complex system, the whole of the system may be greater than the sum of its parts, and the system continuously evolves. Therefore, learning is more suitable for improving system efficiency [9]. Figure 7 illustrates the two kinds of systems. However, to use machine learning or AI to solve system problems, we need to consider how to effectively integrate machine learning with traditional experience-based or system design-based methods to bring the advantages of learned systems into full play.

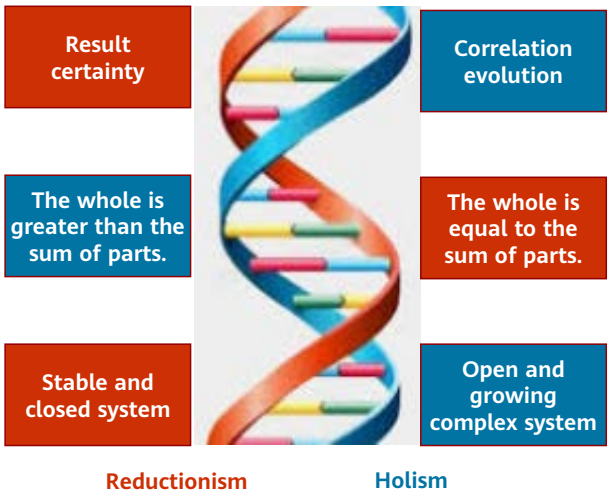


Figure 7 Differences between reductionism and holism [9]

There are many OS parameters and both the service load and system status constantly change. As such, how to implement automatic optimization and O&M is a common challenge in the industry and a research hotspot in academia. In recent years, academia has made progress in this field. Such progress includes learning-based OS [11], learning-based extensible indexing [12], learning-based caching [13], and learning-based concurrency control [14].

4 Prospect of Hardware-driven OS Innovations

4.1 Heterogeneous Many-Core SoCs, XPU Collaboration, and New Computing Architectures Need New Computing Paradigms

From the perspective of computing power, OSs need to collaborate with and promote the evolution of computing

paradigms in heterogeneous many-core systems on a chip (SoCs), XPU collaboration, and new computing architectures, as shown in Figure 8.

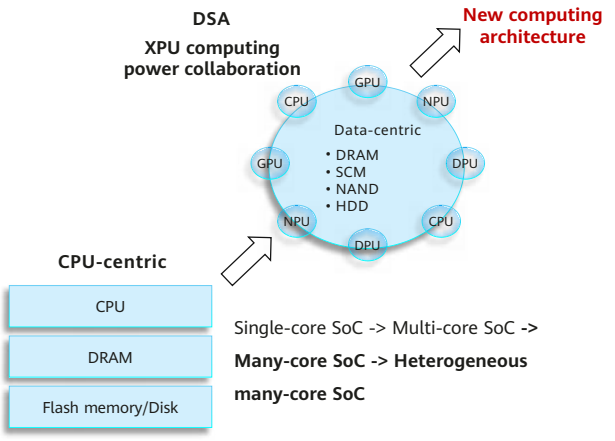


Figure 8 Computing power evolution

- **Heterogeneous many-core SoCs:** CPUs have evolved from single-core CPUs to multi-core CPUs, and then to the current many-core SoCs. In server scenarios, hundreds or even thousands of CPU cores are available. In addition, the big.LITTLE architecture and heterogeneous cores, which used to be applied to terminal devices such as mobile phones, are now also applied to PCs (e.g., Apple has used its M1 chip in desktops, notebooks, and tablets) and may even be utilized on servers in the future. Many challenging topics centered on heterogeneous many-core SoCs need to be researched, including but not limited to: combining massive scalable synchronization primitives [15], improving the reliability and scalability of synchronization primitives to reliably release concurrent computing power [16], non-uniform memory access (NUMA)+asymmetric multiprocessing (AMP) architecture awareness, and transient thread migration.
- **XPU computing collaboration in Domain-Specific Architecture (DSA):** Breakthroughs need to be made in efficient collaboration among XPUs (CPU, GPU, NPU, DPU, etc.) to enable OSs to evolve from individual OSs to a set of collaborative OSs on the SoC, thereby maximizing the overall energy efficiency of the SoC.
- **Exploration of new computing architectures:** The disadvantages of DSA have gradually become apparent. For example, vendors need to maintain multiple hardware architectures, overlapping functions result in waste, software stacks are difficult to share, and inter-XPU coordination and scheduling efficiency cannot be improved significantly. Makimoto's Wave [17] holds that the semiconductor industry changes the direction between customization and standardization roughly every ten years. The industry also expects new computing architectures that differ from DSA. Under new computing architectures, the infrastructure and basic capabilities of traditional OSs will also undergo fundamental transformation.

4.2 New Storage Media and Interconnection Technologies Require New Data Management Paradigms from OSs

Traditional cache, memory, and storage differ greatly in speed and cost. They are CPU-centric and form a multi-layer storage structure. However, non-volatile memory, such as storage class memory (SCM), reduces the latency of storage media from milliseconds to microseconds, as shown in Figure 9. In addition, new high-performance, highly scalable interconnection technologies such as UB are emerging [18], as shown in Figure 10. All these changes pose challenges to traditional data management paradigms.

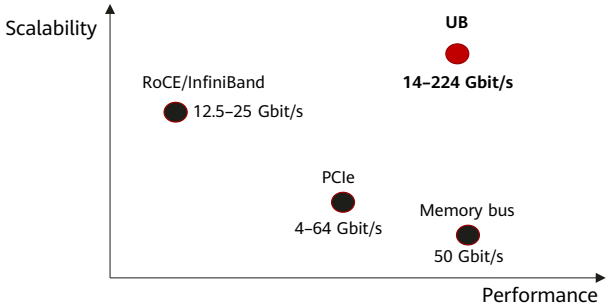


Figure 10 Performance and scalability of interconnection technologies

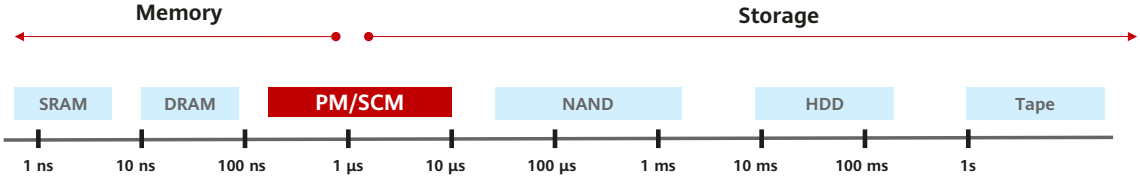


Figure 9 New media are converging memory and storage

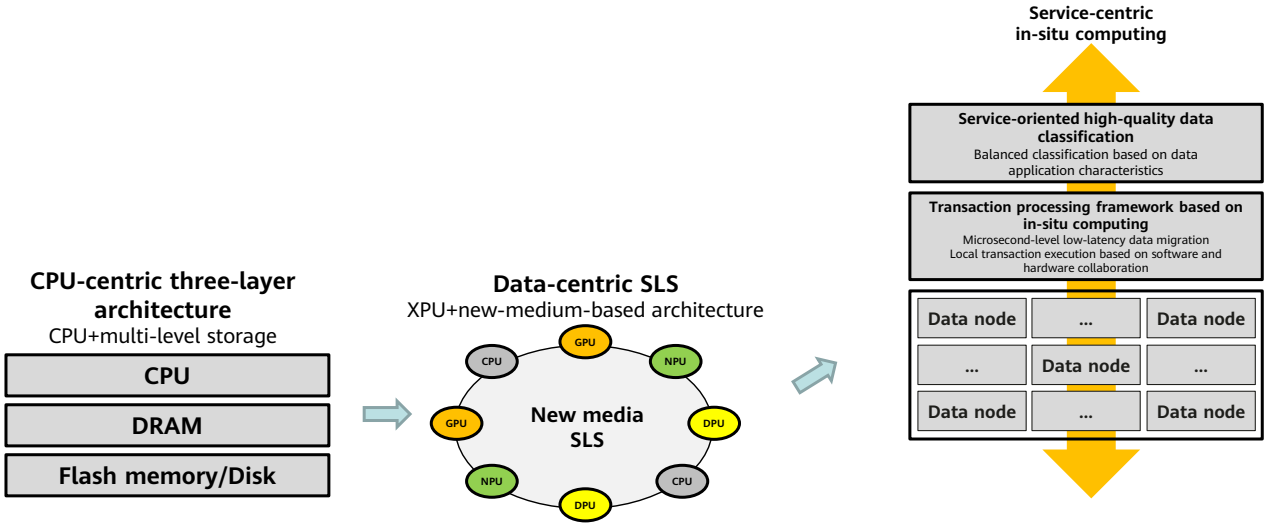


Figure 11 Evolution trend of computing architectures and paradigms of OSs

While the latency of storage media is reduced from milliseconds to microseconds, the proportion of software stack latency in the entire I/O increases exponentially. Therefore, storage software stacks with ultra-low latency [19] and data consistency under high concurrency [20] have become research hotspots in recent years.

Driven by new media and new interconnection technologies, the two-layer (memory+storage) storage management architecture of traditional OSs may gradually evolve toward the data-centric single-level storage (SLS) architecture. That is, memory and storage are managed in an integrated manner at the logic level, greatly reducing data migration. A key question, then, is how to bring data closer to compute nodes to achieve service-centric effects? Near data computing (NDC) systems are needed to answer this question, as shown in Figure 11.

4.3 Confidential Computing Architecture Demands New Trust Paradigms in OSs

Confidential computing is a security technology that protects the code and data of secure computing tasks that run in an environment isolated by using trusted hardware. It has attracted more and more attention in the industry as data security and privacy regulation are continuously strengthened by regulations such as the European Union (EU) General Data Protection Regulation (GDPR) and Data Security Law of the People's Republic of China. Since ARM proposed the TrustZone architecture in 2002, various technical solutions and abstract evolutions of confidential computing have emerged, as shown in Figure 12 [21].

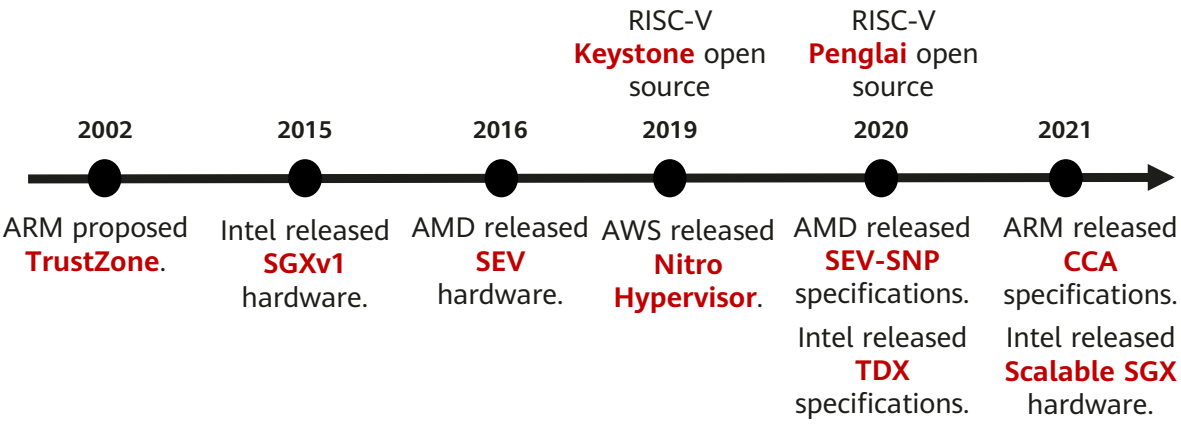


Figure 12 Key technologies and development history of confidential computing

Proposed by ARM in 2002, TrustZone is an isolation mechanism that isolates objects such as CPUs, memory, bus structures, and peripheral devices, and provides independent physical machine abstraction [22]. Currently, TrustZone is widely deployed on ARM-based mobile phones to implement security features such as biometric identification (e.g., fingerprint recognition, facial recognition, and iris recognition) and Android key management. Generally, a small, secure OS, such as Huawei iTrustee [23], OP-TEE [24], and Trustonic [25], runs in ARM TrustZone.

At ARM Vision Day 2021, ARM released its next-generation Armv9 architecture. Confidential compute architecture (CCA), a key component of the Armv9 architecture, introduced the Realm world, which is isolated from both Non-secure World and Secure World (i.e., TEE). CCA essentially transforms the trust paradigm from "trusting devices but not applications" in the device domain into "trusting applications but not necessarily trusting infrastructure providers" in the cloud computing domain. OSs need to support the Realm Manager mechanism and the trusted OS that run on Realm Manager in order to effectively support the trust paradigm transformation brought by ARM CCA [26].

In 2015, Intel officially announced the first-generation Software Guard Extensions (SGX) technology. SGX defines a secure area (named enclave) in the user-mode process. The code and data in the enclave are invisible to the outside. Programmers need to divide the code into the secure part (enclave) and non-secure part (app) in an explicit and static manner and specify the interfaces for interaction between the two parts. Because SGX's security threat model does not trust the OS, SGX defines a number of functions in its hardware design to check the behavior of the underlying untrusted privileged software. These functions include secure page table management, secure thread management, secure memory paging management, and secure memory integrity check. Enclave abstraction is part of the app, and the interfaces and interaction between the app and the OS are extremely complex and constantly evolve. As a result, the hardware design of SGX is complex, hardware needs to carry many software functions, and performance overheads are high. In January 2022, Intel announced that it would abandon all SGX features in its new desktop, notebook, and embedded processors, except for servers [27].

The successor to SGX is Trust Domain Extensions (TDX), Intel's new secure computing abstraction. TDX introduces an independent trusted hypervisor, also called virtual machine monitor (VMM), which shares the same physical host with the original cloud platform VMM. A trusted hypervisor (called the TDX module) is a small secure module that checks the interaction between a trusted VM in the trusted domain (TD) and an external untrusted VMM.

From the perspective of evolution trends represented by ARM CCA and Intel TDX, the evolution of mapping trust paradigms needs to be studied in the OS field to reduce the size of trusted computing base (TCB) in confidential computing abstraction by innovating the architecture design (e.g., microkernel architecture) of the OS. In addition, suitable methods, such as hardware-software collaborative security hardening, formal verification, and more secure languages (e.g., Rust), are required to ensure the security and trustworthiness of TCB in confidential computing abstraction.

5 openEuler and OpenHarmony Practices

5.1 openEuler Practices

In December 2019, Huawei initiated the openEuler project. The aim was to open up its OS capabilities accumulated over the years and to work with industry partners in order to help drive the OS industry. At the openEuler Summit held in November 2021, Huawei joined hands with community partners to officially donate Euler to the OpenAtom Foundation, transforming Euler from an enterprise-led open-source OS to an industry-led one [28].

From the perspective of industry applications, openEuler has built a flexible architecture to support ICT and OT scenarios (servers, cloud computing, edge computing, embedded systems, etc.), and has become a unified open-source OS oriented to digital infrastructure. From the perspective of hardware, openEuler supports collaboration of diversified computing capabilities, builds storage software stacks oriented to new storage media such as SCM, and provides an automatic, intelligent performance optimization engine.

With the joint efforts of multiple parties, openEuler and its derivatives had been applied in more than a dozen industries by March 2022. In addition, openEuler supports certification for more than 5000 types of software, is compatible with more than 60 devices and over 200 mainstream board cards, and supports multiple processor architectures such as Kunpeng, x86, Phytium, Loongson, Sunway, and RISC-V [29].

5.2 OpenHarmony Practices

Since 2020, Huawei has donated the basic capabilities of its smart device OS HarmonyOS to the OpenAtom Foundation, which has integrated the contributions of other participants to form the OpenHarmony open-source project. Huawei continues to help build this project.

OpenHarmony is a unified OS oriented to smart devices in the era of intelligent connection of everything. It provides a unified language for intelligence, interconnection, and collaboration of numerous devices, providing consumers with a simple, direct, smooth, continuous, secure, and reliable all-scenario interaction experience. OpenHarmony has three unique characteristics:

- One OS can meet the requirements of a wide range of devices and be deployed in a unified and elastic manner. OpenHarmony provides scalable hardware resources (with the memory size ranging from several hundred KB to several GB) through component-based and elastic design, and supports multiple processor architectures, including ARM, RISC-V, and x86.
- The devices running OpenHarmony are integrated into a hyper terminal at the system level so that the hardware capabilities of these devices can be flexibly scaled, implementing hardware cooperation and resource sharing among devices. The distribution capability of OpenHarmony enables quick discovery and interconnection among devices, efficient data transmission, and cross-device task coordination.
- OpenHarmony provides a cross-terminal application development framework and supports API consistency, implementing one-time development and multi-device deployment for developers.

The technical architecture of OpenHarmony complies with the hierarchical design philosophy, and consists of the kernel layer, system service layer, framework layer, and application layer from bottom to top. System functions are provided by level as follows: system > subsystem > component. In multi-device deployment scenarios, some non-essential components can be tailored as required.

5.3 Collaborative Innovations Between openEuler and OpenHarmony

openEuler focuses on digital infrastructure, whereas OpenHarmony focuses on smart devices. Collaborative innovations between openEuler and OpenHarmony are expected to enable a collaborative OS that covers cloud, pipe, edge, and device capabilities.

- **Capability sharing between openEuler and OpenHarmony:** The shared capabilities include the kernel, driver framework, collaboration bus, and device-cloud collaborative runtime environment, as shown in Figure 13.

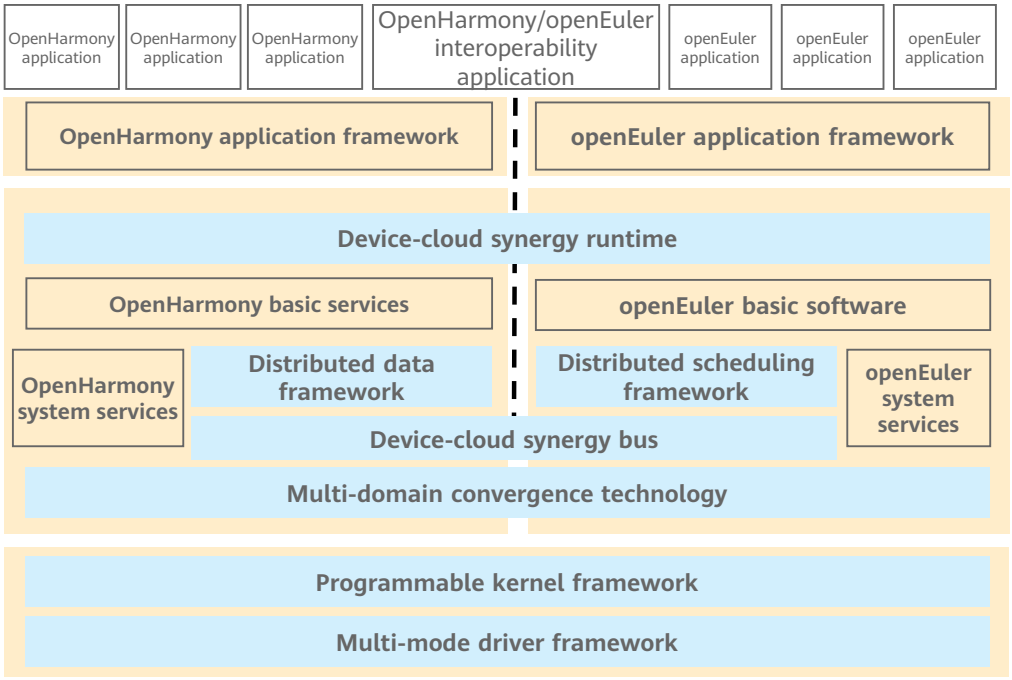


Figure 13 Capability sharing between openEuler and OpenHarmony

- **Multi-domain integration and advantage fusion enabled by streamlining openEuler and OpenHarmony:** In certain complex scenarios, for example, in scenarios where office PCs need to function smoothly, consume minimal power, and leverage existing productivity tools and application ecosystems, openEuler and OpenHarmony can be streamlined to implement hybrid deployment of multiple domains on one device, as shown in Figure 14.

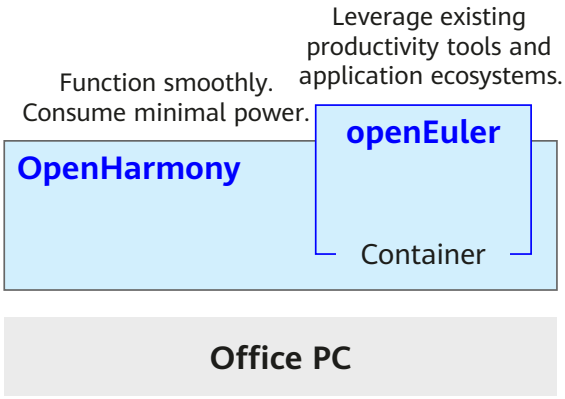


Figure 14 Multi-domain combination of openEuler and OpenHarmony

- **Cloud-network-edge-device cross-domain convergence between openEuler and OpenHarmony:** Based on technologies such as near-field and far-field soft bus convergence, openEuler and OpenHarmony implement capabilities such as self-discovery, self-connection, elastic scaling, and Service Level Agreement (SLA), experience-aware link management, and cloud-network-edge-device coordination of computing power. See Figure 15.

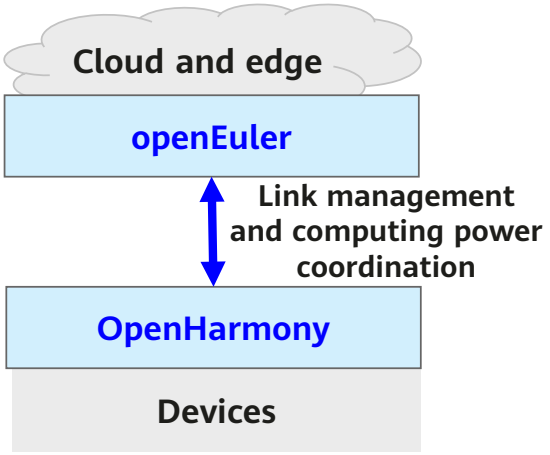


Figure 15 Cloud-network-edge-device cross-domain convergence between openEuler and OpenHarmony

6 Conclusion

OSs serve upper-layer applications, and manage and expose lower-layer hardware capabilities. In this paper, we reviewed the development of OSs from two dimensions: industry and application evolution and hardware evolution.

From the perspective of industry and application evolution, diversified product forms in the era of intelligent connection of everything call for elastic OS architectures. The emergence of new computing forms, such as function computing and edge computing, require new computing abstractions that are ultra-lightweight and securely isolated, and have short life cycles. The contradiction between system complexity and agility will promote the development of learned systems.

From the perspective of hardware evolution, the evolution toward heterogeneous many-core SoCs, XPU collaboration under DSA, and new computing architectures demands new computing paradigms in the OS field. New storage media and interconnection technologies require new data management paradigms. In addition, the CCA in the Armv9 architecture is pushing the OS trust paradigm to transform from "trusting devices but not applications" in the device domain into "trusting applications but not necessarily trusting infrastructure providers" in the cloud computing domain, prompting the emergence of confidential computing OSs.

While openEuler focuses on digital infrastructure, OpenHarmony focuses on smart devices — collaborative innovations between them are expected to enable a collaborative OS that covers cloud, pipe, edge, and device capabilities.

References

[1] Xiaoxiang Zhang *et al.* *Encyclopedia of Computer Science and Technology*, Third Edition, Tsinghua University Press, 2018.

[2] https://en.wikipedia.org/wiki/GM-NAA_I/O

[3] Frederick P. Brooks Jr., "The IBM operating system/360," *Software Pioneers*, Springer, Berlin, Heidelberg, 2002, 170-178.

[4] Frederick P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*, Pearson Education, 1995.

[5] V. A. Vyssotsky, F. J. Corbató, and R. M. Graham, "Structure of the Multics supervisor," *Proceedings of the November 30--December 1*, 1965, Fall Joint Computer Conference, part I, 1965.

[6] Dennis M. Ritchie, "The evolution of the Unix time-sharing system," *Symposium on Language Design and Programming Methodology*, Springer, Berlin, Heidelberg, 1979.

[7] David C. Sastry and Mettin Demirci, "The QNX operating system," *Computer*, 28.11 (1995): 75-77.

[8] https://www.windriver.com/products/vxworks

[9] Huaimin Wang, "Distributed computing 2.0: Network-based connected computing," *Invited Speech at 2020 CCF ChinaSoft*, 2020.

[10] Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff, "Architectural implications of function-as-a-service computing," *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019.

[11] Yiying Zhang and Yutong Huang, "Learned operating systems," *ACM SIGOPS Operating Systems Review*, 53.1 (2019): 40-45.

[12] Chuzhe Tang *et al.*, "XIndex: A scalable learned index for multicore data storage," *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2020.

[13] Xingda Wei, Rong Chen, and Haibo Chen, "Fast RDMA-based ordered key-value store using remote learned cache," *14th USENIX Symposium on Operating Systems Design and Implementation*, Banff, Alberta, Canada, November 2020.

[14] Jiachen Wang, Ding Ding, Huan Wang, Conrad Christensen, Zhaoguo Wang, Haibo Chen, and Jinyang Li, "Polyjuice: High-performance transactions via learned concurrency control," in *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation*, July 2021.

[15] Rafael Lourenco de Lima Chehab *et al.*, "CLoF: A compositional lock framework for multi-level NUMA systems," *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021.

[16] Jonas Oberhauser *et al.*, "VSync: push-button verification and optimization for synchronization primitives on weak memory models," *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021.

[17] Tsugio Makimoto, "Implications of Makimoto's Wave," *Computer*, 46.12 (2013): 32-37.

[18] Tan Kun *et al.*, "UB: Unified and scalable memory-semantic interconnection for next-generation computing systems," *Communications of HUAWEI RESEARCH*, vol. 1.

[19] Mingkai Dong and Haibo Chen, "Soft updates made simple and fast on non-volatile memory," *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017.

[20] Jifei Yi *et al.*, "HTMFS: Strong consistency comes for free with hardware transactional memory in persistent memory file systems," *Proceedings of the 20th USENIX Conference on File and Storage Technologies*, 2022.

[21] Fahmida Y. Rashid, "The rise of confidential computing: Big tech companies are adopting a new security model to protect data while it's in use-[news]," *IEEE Spectrum*, 57.6 (2020): 8-9.

[22] Sandro Pinto and Nuno Santos, "Demystifying Arm TrustZone: A comprehensive survey," ACM Computing Surveys (CSUR), 51.6 (2019): 1-36.

[23] https://developer.huawei.com/consumer/cn/forum/topic/0202513076169910329?fid=0102501639476130680&postId=0302513076169910363

[24] https://www.op-tee.org/

[25] https://www.trustonic.com/

[26] Dominic P. Mulligan *et al.*, "Confidential computing—a brave new world," *2021 IEEE International Symposium on Secure and Private Execution Environment Design (SEED)*, 2021.

[27] https://community.intel.com/t5/Blogs/Products-and-Solutions/Security/Rising-to-the-Challenge-Data-Security-with-Intel-Confidential/post/1353141

[28] https://openeuler.org/

[29] https://www.openeuler.org/zh/compatibility/

GaussDB: Cloud-Native Distributed Database

Andy Li, Yang Ren
Gauss Dept

Abstract

Huawei GaussDB is an enterprise-grade distributed database kernel developed by Gauss Department (under Central Software Institute of the 2012 Laboratories). It started as a prototype more than ten years ago and is now being widely used by enterprise customers around China, including large-scale financial institutions. Five of China's top seven banks have chosen GaussDB for their core business scenarios, including core banking, internet finance services, channel business, and customer relationship management (CRM)/enterprise resource planning (ERP). By the end of 2021, GaussDB had served more than 1500 enterprise customers in China. This article describes the architecture and some major features of GaussDB. GaussDB has the following competitive advantages: high availability, high performance, hybrid workload, security, and autonomy.

Keywords

distributed database, cloudification, high performance, high availability, hybrid workload, security, autonomy, openGauss, GaussDB

1 Introduction

As new services and infrastructures emerge, applications pose new technical requirements for databases. The popularization of the mobile internet is resulting in a large number of 2C services. Financial services, e-commerce applications on mobile devices, social applications, and Internet of Things (IoT) devices are generating various types of massive data. All these changes require a transformation in the traditional over-the-counter and centralized processing-based service models. Against this backdrop, databases have evolved from centralized databases to distributed databases, and then to cloud-based distributed databases. To meet the requirements of finance services, enterprise services, and related applications, and efficiently utilize cloud computing infrastructure, Huawei launched GaussDB, a self-developed, enterprise-grade, cloud-based distributed database. Different from databases that use the traditional distributed application architecture and distributed middleware architecture, GaussDB is designed to meet distribution requirements right from the start, and has a built-in Structured Query Language (SQL) engine, execution engine, and storage engine. GaussDB has the following competitive advantages: high availability, high performance, hybrid workload, security, and autonomy. GaussDB adopts the shared-nothing massively parallel processing (MPP) architecture and complies with the American National Standards Institute (ANSI) SQL 2008 standard.

The rest of this article is organized as follows. Section 2 presents an overview of the GaussDB architecture and the competitiveness of GaussDB in five dimensions. In section 3, we summarize our work and discuss future work.

2 Architecture

This section first presents the architecture of GaussDB.

GaussDB is designed based on the shared-nothing architecture. A single set of GaussDB can be linearly scaled to hundreds of physical machines and handle various workloads in parallel. Database data is partitioned and stored in data nodes (DNs) according to certain conditions. The DNs meet the local atomicity, consistency, isolation, durability (ACID) property requirements. The system uses the two phase commit (2PC) mechanism and global transaction manager (GTM) to ensure cross-node consistency. GaussDB supports both row stores and column stores. Based on column store principles, GaussDB implements a vectorized execution engine to efficiently utilize the latest single instruction multiple data (SIMD) instructions of the processor, thereby achieving instruction-level fine-grained parallelism. The system-generated execution plans and executors are designed for distributed systems. The execution plans can be processed in parallel on hundreds of servers. Data is exchanged on demand across nodes and queries are executed in parallel. To minimize the performance deterioration of cross-node transactions, GaussDB uses the timestamp-based GTM-Lite technology. Logical instances of GaussDB include the operation manager (OM), cluster manager (CM), GTM, coordinator nodes (CNs), and DNs. An application sends an SQL statement to a CN, which compiles the statement into an execution plan and pushes the plan to the related DNs for query execution. The DNs aggregate the results to the CN, which returns the aggregation result to the application.

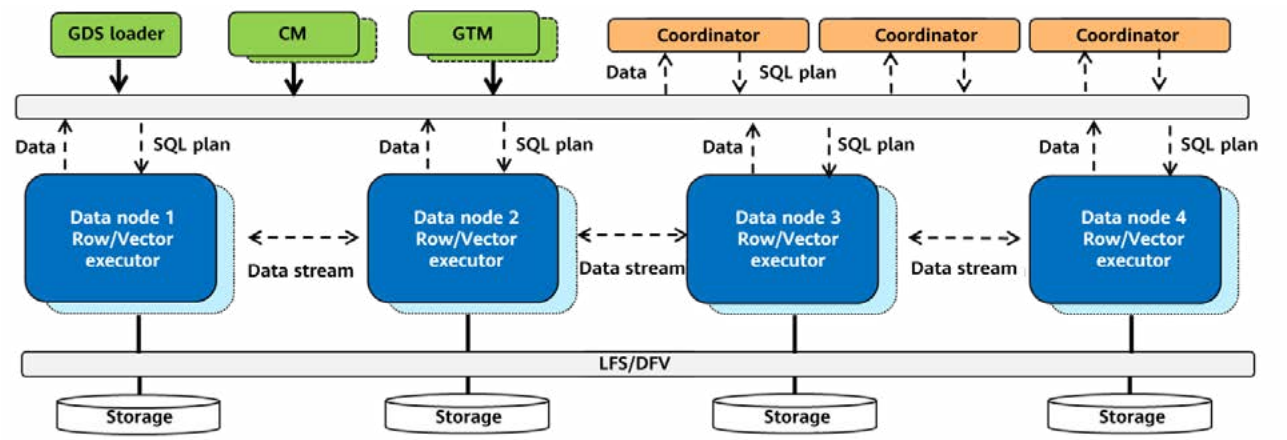


Figure 1 High-level logical system architecture of GaussDB

Table 1 System instance description

Instance	Function
CN	Receives SQL statements, generates execution plans, and coordinates DNs to complete execution plans.
DN	Stores data. DNs collaborate with each other to complete SQL queries.
GTM	Generates the global timestamp and provides global snapshots.
CM	Consists of CM Agent and CM Server, handles the cluster quorum, and provides high availability capabilities.
Gauss Data Source (GDS)	Implements parallel data loading.

2.1 High Performance

GaussDB fully leverages hardware resources to deliver high performance. It reconstructs key data structures in the database kernel for Kunpeng processors, preventing cross-chip access of global shared data structures. As a result, it greatly reduces false sharing of CPUs. In addition, GaussDB uses the atomic Load Effective Address (LEA) instructions produced by Kunpeng processors to refactor the spin lock in the database, improving the spin lock performance by four times. Based on the storage class memory (SCM) and remote direct memory access (RDMA) technologies, the GaussDB team researched on memory-centric architectures and the next-generation distributed database communication network. All these factors enable GaussDB to provide a processing capability of 1.5 million tpmC on a single-node 2-socket Kunpeng server, and a processing capability of over 2.3 million tpmC on a single-node 4-socket Kunpeng server, with a linear scaling ratio of 1.5.

In addition, GaussDB enhances system performance in the following four aspects:

- Shared-nothing architecture: GaussDB distributes data to DNs based on certain rules. When executing a query, a DN processes its own data locally and shuffles data across nodes as required.
- Symmetric multiprocessing (SMP) technology: To fully exploit the multi-core capabilities of modern CPUs, GaussDB adopts the SMP technology for query processing. Within a node, data is split into multiple pieces and processed in parallel by multiple threads. In some compute-intensive queries, the system fully utilizes the CPU to accelerate query processing.
- Instruction-level parallelism (ILP): Implemented by GaussDB based on the SIMD capability of modern CPUs, the vectorized execution engine processes data in batch mode (1000 tuples per batch), enabling the system to achieve high scores in the TPC-H and TPC-DS benchmark tests.
- Dynamic compilation technology: The dynamic compilation policy is used to continuously improve processing performance. The Low Level Virtual Machine (LLVM) technology is used to reduce the number of CPU instructions involved in queries, thereby improving query performance.

2.2 High Availability

The high availability capability of GaussDB consists of five parts: fault prevention, fault tolerance, fault correction, fast locating, and serviceability. Since its inception, the GaussDB architecture is designed to prevent single point of failure (SPOF) in the system, and hardware components such as switches, network adapters, and power supplies are configured in redundancy mode. For local disks, the system uses RAID 5 to protect data and prevent service interruption if data corruption occurs on a single disk. For EVS disks on Huawei Cloud, the system uses their multi-copy mechanism to provide data availability. In addition to hardware redundancy, the system also uses the high availability architecture for instance roles. In the system, CNs use the multi-copy peer-to-peer architecture, DNs use the primary/standby copy mechanism, and the CM and GTM use the warm backup mechanism. The CM performs primary/standby arbitration for each role, whereas the Editable Text Configuration Daemon (ETCD) performs primary/standby arbitration for the CM. Serving as the brain of a cluster, the CM determines the primary/standby role of each instance. The CM itself has multiple warm backups. When the primary instance of the primary CM in the system is faulty, the system selects a standby CM and promotes it to primary to take over the cluster quorum after arbitration by the ETCD. The primary instances and warm backup instances of roles such as the GTM and DNs periodically send heartbeat messages to the primary CM instance. The CM determines the primary instance of each role based on the received heartbeat messages and performs a switchover if the primary instance becomes faulty. In addition, GaussDB provides various types of log analysis tools, core dump analysis tools, and distributed tracing tools to efficiently

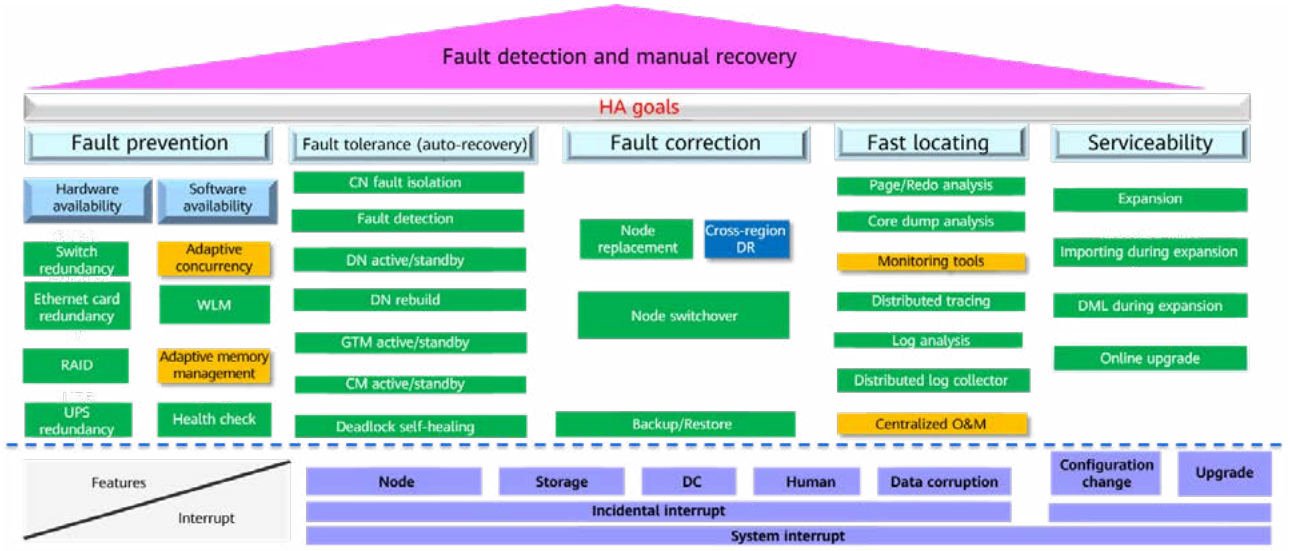


Figure 2 High availability capability panorama of GaussDB

locate system faults. GaussDB also supports in-service O&M. The online upgrade and online scaling functions ensure that service work flows are not interrupted during the upgrade and scaling operations.

GaussDB provides multi-level high availability capabilities, such as single availability zone (single-AZ) high availability, cross-AZ high availability, and cross-region high availability, for production systems used in the financial industry, based on the specific high availability requirements of services. In an AZ, GaussDB provides the high availability capability (RPO = 0 and RTO < 10s) based on the no-SPOF design. In addition to ensuring AZ-level high availability, GaussDB can provide an ultra-high performance of 120 million tpmC on more than 100 physical nodes. GaussDB with cross-AZ cluster deployment provides strong consistency and high availability (RPO = 0 and RTO < 30s). If a service needs to provide cross-region high availability for an area with a radius greater than 1000 km, GaussDB can ensure an RPO less than 10s and an RTO within minutes. To meet high availability requirements in larger geographic areas, for example, an area with a radius greater than 2000 km, GaussDB provides the global database capability based on global clock consistency and provides 99.999% availability for the entire system. To sum up, GaussDB can meet the multi-level high availability capability requirements of different applications while ensuring high performance.

2.3 Hybrid Workload

One of the standard capabilities of an enterprise-grade database is to support both online transaction processing (OLTP) and online analytical processing (OLAP) workloads. Mainstream databases such as Oracle and SQL Server have such a capability. It is difficult to classify the workload of an enterprise-grade service as OLTP only or OLAP only. The hybrid workload capability is essential for enterprise-grade databases. GaussDB greatly improves the complex query capability of the database by using the fully parallel architecture, including inter-node parallelism, intra-node parallelism, and instruction parallelism. GaussDB also uses the enterprise-grade distributed optimizer to implement functions such as view extension, parameterized path, and lazy aggregation. In the same hardware environment and test set, the distributed GaussDB database consumes 82% less time than other similar products, for complex queries. Based on the powerful complex query capability, the lightweight global snapshot GTM-Lite technology greatly improves the transaction processing performance of the system. The system performance increases linearly as the number of nodes increases. In actual applications, the number of nodes in the largest GaussDB cluster is approximately 1000, and the number of vCPU cores consumed by the cluster is nearly 100,000.

The performance of hybrid workloads is reliant on the availability of system resources. GaussDB depends on the global workload manager to control the number of concurrent queries. The workload manager optimizes the system throughput and prevents processing performance degradation caused by queries competing for system resources. The workload manager consists of three major parts: resource pool, workload group, and controller. The resource pool is used to allocate shared system resources,

such as memory and disk I/O, for queries running in the system, and to set various execution thresholds in order to determine the query execution modes. All queries run in the same resource pool, and the workload group is used to assign the arrived queries to the resource pool. The controller evaluates queries and dynamically makes execution decisions based on the resource requirements (i.e., cost) of the queries and the available system resources (i.e., capacity). If the estimated cost of a query is lower than the available capacity of the system, the query is executed. Otherwise, the query is queued. Resource recording and feedback mechanisms are used to track the available capacity of the system. When the available capacity of the system is sufficient, the query is de-queued and then executed by the execution engine.

2.4 Security

On the cloud, the distributed GaussDB database provides comprehensive security protection for customer data during data transmission, query processing, computing, O&M, and idle periods. To ensure data security during query processing, GaussDB provides the fully encrypted query function. Data is processed in the encrypted state to complete query requests and is decrypted only when data is returned to the client, ensuring data security during transmission and query processing. GaussDB uses trace chains to record all database changes and identify historical data changes during O&M. The system automatically detects and corrects tampered data if multiple parties are involved.

Due to the open environment and blurred network boundaries, cloud-based databases face more diverse and serious threats than ever before. Although most cloud-based database systems employ various protection mechanisms, such as data encryption, identity authentication, and auditing, these protection mechanisms are based on the same assumption: database users trust the cloud infrastructure and privileged users (e.g., the administrator). However, this assumption is fragile to a certain extent, and puts users' sensitive data at great risk. Given this, database systems generally adopt end-to-end defense mechanisms, such as fully homomorphic encryption and property-preserving encryption. Microsoft Azure SQL Database allows data owners to encrypt data at column-level granularity and utilize the trusted execution environment (TEE) to securely perform complex operations on ciphertext data. Therefore, even privileged users or infrastructure providers cannot gain access to users' sensitive data. However, protection mechanisms such as this pose challenges to database design and implementation. To tackle such challenges, a novel security mechanism called full encryption in GaussDB (FE-in-GaussDB) is proposed to protect data in both private and public cloud environments. FE-in-GaussDB combines the strengths of software and hardware modes and related security functions to achieve processing flexibility. The software mode includes data encryption and indexing schemes for efficient equality query and range comparison operations on ciphertext data, and runs in the rich execution environment (REE), i.e., the Normal World. The hardware mode leverages the hardware TEE (i.e., the Secure World) provided by the chip to securely handle the decryption and complex computations, such as string search operations and the aggregation function, on ciphertext data. FE-in-GaussDB supports full-scenario SQL query processing and has the following advantages:

- Novel database security capability pattern that combines the existing data encryption scheme of the software mode and the TEE-based secure computation of the hardware mode.
- Full-lifecycle encryption, making FE-in-GaussDB transparent to database users and applications.
- Database-based execution mode that strikes a balance between security and performance.
- To implement the FE-in-GaussDB security mechanism, GaussDB modifies the optimizer and executor as follows:
- The optimizer is modified to determine whether the request is relevant to the cipher fields and whether it should be processed within the TEE.
- A two-level cipher index is built. Upper-layer indexes use labels to represent different processing ranges. Lower-layer indexes use the TEE to maintain sequential storage.
- Data definition language (DDL) operations and data manipulation language (DML) statements with equality query classes are directly processed in the database using the software mode without accessing the TEE.
- Both the software mode and hardware mode use the same client encryption driver and client parser module.

2.5 Autonomy

An autonomous database framework is built for GaussDB and integrated into openGauss. This framework consists of the following five parts:

- Learned optimizer: GaussDB proposes the learned query rewriter, learned cost/cardinality estimator, and learned plan generator (including join order selection and physical operator selection). The learned query rewriter uses a Monte Carlo tree search-based method to rewrite an SQL query into an equivalent, yet more efficient query by considering the benefits and orders of rewrite. The learned cost/cardinality estimator uses tree long short-term memory (Tree-LSTM) to simultaneously estimate cost and cardinality. The learned plan generator utilizes deep reinforcement learning (RL) to select a good join order and appropriate physical operators.
- Learned database self-monitoring and self-diagnosis: GaussDB uses learning-based self-monitoring, self-diagnosis, self-configuration, and self-optimization techniques to monitor, diagnose, configure, and optimize databases. Self-monitoring tracks database metrics and uses the metrics to optimize the operational behavior of other components. Self-diagnosis uses Tree-LSTM to detect anomalies and identify the root causes of anomalies. Self-configuration uses deep RL to tune knobs. Self-optimization uses an encoder-decoder model to recommend views and a deep RL model to recommend indexes.
- Model validation: To validate whether a learned model is effective for workloads, the GaussDB team proposes a graph embedding-based performance prediction model. The system predicts the performance of models in the learned optimizer or learned advisor prior to model deployment. If the performance is predicted to improve, the model is deployed. Otherwise, the model is discarded.
- Model management: Most models in the learned optimizer and learned advisor are realized by combining multiple RL or deep learning algorithms. The GaussDB team provides a machine learning (ML) platform to implement unified resource scheduling and model management. This platform encapsulates the complexity of ML and provides developer-oriented training, prediction, and model management capabilities.
- Training data management: GaussDB collects runtime database metrics (e.g., resource consumption and lock/latch information), historical SQL queries (e.g., query latency), and system logs (e.g., anomalies) as the raw input data for model training, and organizes and manages the collected data as follows:
  - Judiciously organizes correlated columns into the same table to reduce join overhead
  - Judiciously selects training data to train a learned model

To improve system O&M capabilities, GaussDB leverages AI capabilities such as deep RL and graph neural network (GNN) to implement multiple functions, including tuner, system diagnosis, materialized view (MV) recommendation, and index recommendation, greatly improving system efficiency. Take the index recommendation function used by one of our customers as an example. After the customer used this function, the index redundancy was reduced by 83% and the occupied disk space was reduced by 70%, while improving the performance of the customer's production system.

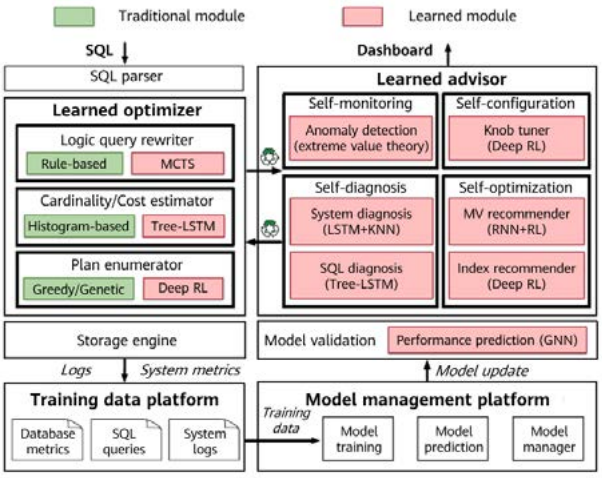


Figure 3 Logical architecture of the autonomous GaussDB database

3 Conclusion

We have conducted extensive research focused on database innovations. In recent years, we have cooperated with academic institutions, including Tsinghua University, and published more than 50 academic papers on Special Interest Group on Management of Data (SIGMOD), Very Large Data Base (VLDB) conference, and International Conference on Data Engineering (ICDE). The paper topics cover AI native databases, transaction processing, in-memory databases (IMDBs), and distributed database systems. In the future, we will deepen our cooperation with academia around the world and continue to focus on database innovation. Faced with the emergence of new application scenarios, new hardware architectures, and new technologies, we will also continue our exploratory research on in-depth collaboration between software and hardware, extreme scalability, serverless, and multi-model convergence. We will also continue to help make the openGauss community mainstream by promoting its development.

Open-Source Ecosystem of AI Systems: Experience on MindSpore

Fan Yu, Beiji Shi, Zidong Wang, Xuefeng Jin, Lei Chen, Teng Su, Ruifeng Li, Bin Zhou, Cheng Ding, Kun Tan

Abstract

MindSpore is an all-new deep learning framework that aims to enable easy development, efficient execution, and adaptability to all scenarios. To improve the ease of development, MindSpore leverages an auto differentiation (AD) mechanism based on source code transformation (SCT), where complex combinations are represented using control flows. With intermediate representations (IRs), functions are transformed into computational graphs that can be analyzed and executed on various devices. To improve the runtime performance and efficiency in device, edge, and cloud scenarios, several hardware-software co-optimization techniques (e.g., graph kernel fusion) are introduced to these graphs prior to their execution. Furthermore, MindSpore supports easy switching between dynamic and static graphs. To support effective training of foundation models on large datasets, MindSpore offers three flexible training approaches: data parallelism, model parallelism, and hybrid parallelism. In addition, MindSpore supports auto parallelism, which effectively searches in the strategy space for a fast parallelism strategy. MindSpore is compatible with a wide array of AI methods such as convolutional neural networks (CNNs) and graph neural networks (GNNs), giving it a distinct competitive edge over open-source counterparts.

Thanks to the exceptional performance that MindSpore delivers on Huawei Ascend AI processors, its open-source ecosystem is booming. And with ever-growing attention from universities, enterprises, and communities, MindSpore has accumulated extensive experience in academic research, business promotion, and community operating practices. Technologies rooted in the MindSpore ecosystem are implemented in more and more enterprises and AI computing centers.

In this paper, we introduce the architecture and core technologies of MindSpore, and discuss our exploration in building an open-source AI ecosystem.

Keywords

deep learning framework, auto differentiation, auto parallelism, graph kernel fusion, dynamic/static graph unification, all-scenario, foundation model, ecosystem building

1 Introduction

Over the past few decades, there has been an explosive growth of deep learning research, along with successful applications in fields such as image recognition [1], speech recognition and synthesis [2], gaming [3], and language modeling and analysis [4]. The continuous development of deep learning frameworks [5–10] opens up access to computation resources in training neural networks on large datasets.

Current deep learning frameworks fall into two primary types. One, represented by TensorFlow [5], builds a static graph that defines each operation and the network structure prior to runtime. Simply put, this approach trades ease of use for training performance. The other, represented by PyTorch [6], uses dynamic computational graphs, which are dynamically built at runtime. While this approach offers flexibility and easier debugging, it does so at the expense of performance. Therefore, existing deep learning frameworks tend to make a trade-off between easy development and efficient execution.

In this paper, we introduce MindSpore — an innovative deep learning framework developed by Huawei — to achieve three goals: easy development, efficient execution, and adaptability to all scenarios. MindSpore consists of several major components, namely, MindExpression, MindCompiler, MindData, MindRE, and MindArmour. Table 1 shows the level of technical contributions these components make to MindSpore's intended goals.

1. MindExpression supports Python, and MindCompiler provides the IR-based just-in-time (JIT) compilation and optimization capability. The two components have the following features:
   - Auto differentiation (AD): The source code transformation (SCT)-based AD mechanism is applied to convert a Python code snippet into a data flow graph for training or inference. In this way, users can easily build sophisticated neural network models using native control logic in Python.
   - Auto parallelism: As models and datasets become larger, parallelizing deep neural network (DNN) training across distributed devices has become a common practice. Unfortunately, existing frameworks such as TensorFlow, Caffe, and MXNet only offer simple and often suboptimal parallel training strategies. MindSpore, however, parallelizes training tasks in a transparent and efficient manner. Transparency is possible because the same Python implementation can adapt to different devices, with only one line of code needing to be reconfigured. And efficiency is achieved by selecting the optimal parallelism strategy with minimum cost, thereby reducing possible computing and communication overheads.
   - Dynamic graph: MindSpore supports dynamic graphs without involving additional AD mechanisms (e.g., operator overloading AD). This significantly improves MindSpore's compatibility with both dynamic and static graphs.
2. MindData is responsible for data processing and implements high-performance data pipelines using the auto data acceleration technology. With various auto augmentation schemes predefined in MindData, developers no longer need to search for appropriate data augmentation strategies. MindData also provides debugging and optimization tools, including a training dashboard (which visualizes the whole training process with all datasets displayed in a centralized manner) and

Table 1 MindSpore's components and their contributions to the three goals

Component / Goal	MindExpression & MindCompiler				MindRE	MindData			
	SCT AD	Dynamic graph	Auto parallelism	Graph manager	Runtime system	Training dashboard	Profiler	Auto augmentation	Auto data acceleration
Easy development	✓	✓	✓	○	○	✓	○	✓	○
Efficient execution	✓	○	✓	✓	○	○	✓	○	✓
Adaptability to all scenarios	○	○	○	✓	✓	○	○	○	○

Note: The ✓ symbol indicates major contribution and the ○ symbol indicates minor contribution.

a profiler (which facilitates performance optimization by measuring the execution time and memory usage statistics provided by the runtime blackbox).

3. MindArmour provides tools for defending against adversarial attacks to achieve privacy-preserving machine learning. MindArmour generates adversarial code, evaluates the performance of models in specific adversarial settings, and develops robust models while also providing ample capabilities for privacy protection.

4. MindRE is the AI network executer. It abstracts away the interface details in low-level hardware and is compatible with various runtime systems in device and cloud environments.

MindSpore is compatible with models developed under all popular deep learning frameworks (such as TensorFlow, PyTorch, and MXNet) and supports full-stack collaborative development across the device, edge, and cloud. This significantly reduces professional requirements and model development time for developers. By supporting AI computing on local hardware, MindSpore eliminates the industry's main concerns about privacy and security. MindSpore has attracted wide attention around the world and created the most active open-source community in China since becoming open source a year ago. Furthermore, MindSpore has a booming open-source ecosystem: it has hit 650,000 downloads, attracted 170,000 developers, and brought over 2000 apps online. In addition, MindSpore has been widely applied in 8 industries, taught in more than 100 universities, attracted contributions from more than 1500 core developers, and been the AI framework used in more than 300 papers published at top-level conferences.

The rest of this paper is organized as follows: Section 2 provides an overview of the MindSpore architecture. Section 3 describes the framework's major components by illustrating the design details of its innovative technologies such as auto differentiation, auto parallelism, dynamic graphs, and graph kernel fusion, as well as MindSpore's extension in GNN applications. Section 4 shares MindSpore's success stories in building an open-source ecosystem and accomplishments of typical projects. Section 5 concludes this work and suggests possible directions for future research.

2 MindSpore Overview

2.1 MindSpore Architecture

Figure 1 shows the architecture of MindSpore.

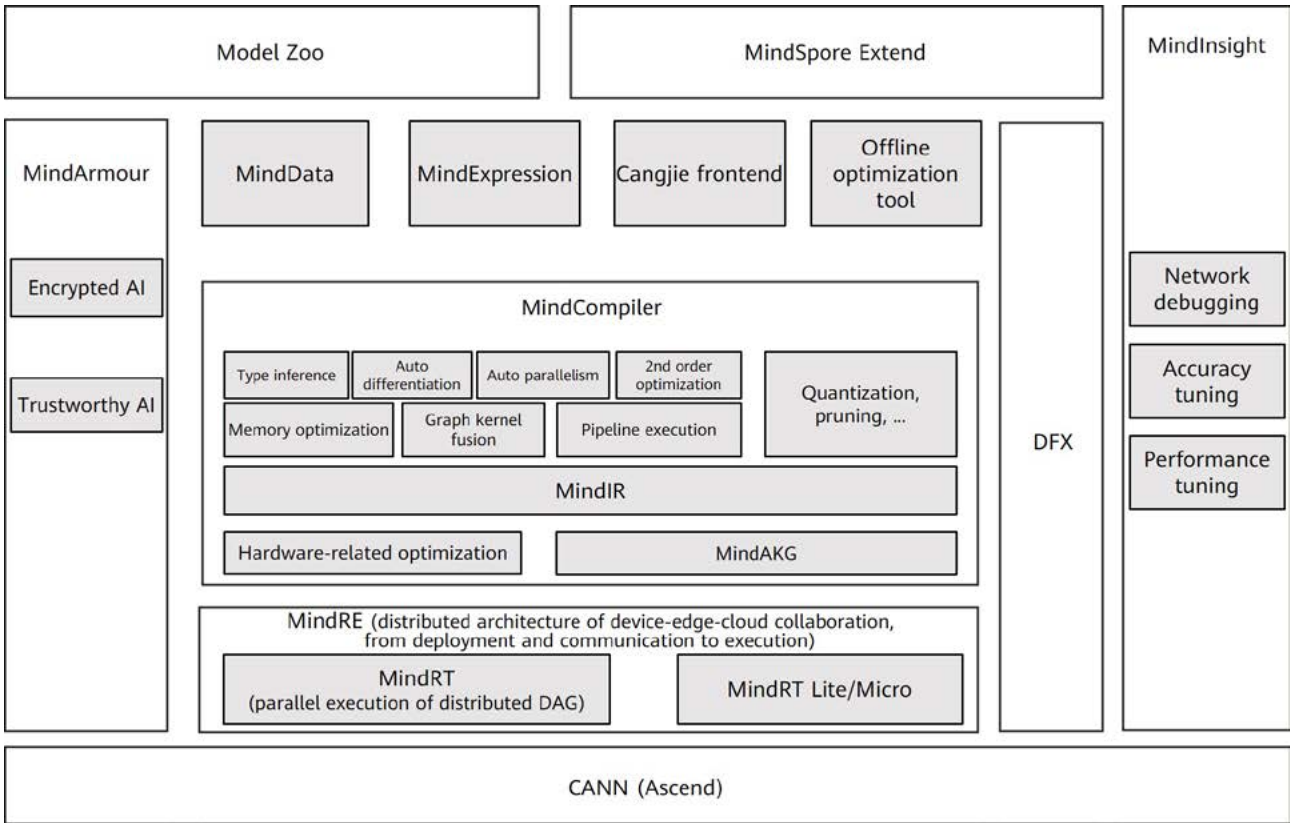


Figure 1 MindSpore architecture

MindExpression provides a Python interface that enables developers to define user-level application programming interfaces (APIs) for building and training neural networks. And thanks to the SCT-based AD mechanism, programming in Python is easy.

MindCompiler is the core of high-performance network execution and SCT-based AD. Networks can run in either PyNative or graph mode. In PyNative mode, network operators are executed one by one. In graph mode, a pipeline is used to build a computational graph from Python code. Specifically, MindCompiler generates an abstract syntax tree by parsing the Python code and then converts the syntax tree into an A-normal-form (ANF) graph. Because ANFs are graphical rather than syntactic, they are easier to manipulate algorithmically. For network training, the pipeline automatically generates backward computation nodes and adds them to the ANF. In addition, the pipeline introduces several beneficial optimizations to the final graph, including memory reuse, operator fusion, and constant elimination. In distributed training scenarios, the pipeline applies the optimization strategy provided by auto parallelism. The back-end virtual machines run graphs on the back-end and control the lifetime of each graph using sessions.

MindData completes data processing pipelines in the training process, including data loading, argumentation, and transformation. It also provides easy-to-use APIs and supports data processing in a wide range of applications such as computer vision (CV), natural language processing (NLP), and graph neural networks (GNNs).

MindInsight provides the training dashboard, lineage visualization, profiler, and debugger. The profiler taps into the runtime blackbox to monitor the execution time and memory usage. The debugger visualizes the internal structure and input/output nodes of a training graph in graph execution mode. MindInsight analyzes training logs, providing insight into the training process.

MindArmour allows developers to build robust models that protect user privacy during model training and inference. It provides three modules for adversarial attack defense: attack, defense, and evaluation. The attack module produces adversarial examples under blackbox and whitebox attack settings. The defense module uses the adversarial examples received from the attack module to improve model robustness during training. And the evaluation module provides different types of evaluation metrics for measuring the robustness of a model in a visualized manner. To achieve privacy-preserving machine learning, MindArmour implements a collection of differential privacy-aware optimizers that automatically add noise to the gradients generated at training time.

2.2 Programming Paradigm

Thanks to the SCT-based AD mechanism, MindSpore supports Python along with control statements native in Python and advanced APIs such as tuples, lists, and lambda expressions.

To maintain simplicity, MindSpore introduces as few interfaces and concepts as possible. For example, to train a simple neural network on a single-node platform, you need to understand only the following five basic components:

- Tensor: A tensor is a multi-dimensional matrix containing elements of a uniform data type. Tensors are convertible to NumPy objects. Unlike existing training frameworks, MindSpore does not have the concept of scalar variables. Instead, MindSpore determines whether to calculate the derivative of a tensor based on the auto differentiation attribute of the tensor.
- Dataset: A dataset is a separate asynchronous pipeline that prepares tensors to feed into the rest of the network with no delay in training.
- Operator: Operators are the basic computational unit of a neural network. In addition to the wide support of popular neural network operators (e.g., convolution, batch normalization, and activation) and math operators (e.g., add and multiply), MindSpore allows developers to create custom operators to adapt to specific hardware platforms and fuse a series of operators into a composite one.
- Cell: A cell is the base class for all neural network cells. Cells are a collection of tensors and operators and can be nested into a tree structure. The compute logic of a neural network is defined using the **construct** function within a cell. The defined compute operations are executed upon each call to this function.
- Model: A model is a high-level API that makes it easy to get started with inference and training on MindSpore. This component is optional, especially for those with knowledge of low-level APIs and who need fine-grained control over how computations are performed.

From the developer perspective, creating a program in MindSpore is to create a cell for a neural network. The

process begins with defining input tensors, followed by building a cell using appropriate operators. It ends with either encapsulating the cell into a model for training or directly feeding the input tensors into the cell for inference.

3 Core Components and Technology Innovations of MindSpore

3.1 MindExpression and MindCompiler

3.1.1 SCT-based AD

Unlike TensorFlow and PyTorch, MindSpore does not implement AD based on static or dynamic graphs. Instead, MindSpore adopts an SCT-based approach. Evolved from functional programming frameworks, this approach performs AD transformation on the IR (the data structure used by the compiler to represent a program in the compilation process) in JIT mode, as shown in Figure 2. AD of the control flow is supported, making it easy to build models like PyTorch. In addition, MindSpore performs static compiler optimization on neural networks, achieving performance as high as TensorFlow does. This novel approach avoids the performance optimization difficulty with dynamic graphs and the complexity involved in network building and debugging with static graphs.

MindSpore's implementation of AD can be understood as the symbolic differentiation of a program itself. MindSpore produces functional IRs that correspond intuitively to composite functions in basic algebra. As long as the derivative formula of a basic function is known, that of the compound function composed of arbitrary basic functions can be deduced. Each primitive operation in an IR corresponds to a basic function in basic algebra, and such basic functions can be used to build control flows with greater complexity.

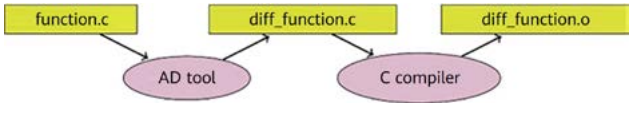


Figure 2 SCT-based AD

3.1.2 Auto Parallelism

As deep learning continues to advance, training datasets and DNN models are growing larger to support higher accuracy and wider applications. In order to train foundation models on large datasets, deep learning frameworks must therefore support not only data parallelism and model parallelism, but also a hybrid of both. Most of today's mainstream frameworks (such as TensorFlow [5], Caffe [7], and MXNet [8]) require DNN models to be manually partitioned in order to implement model parallelism. However, manual partitioning is a complex task and relies on expert experience. Implementing hybrid parallelism of data and the model at the same time adds significant development complexity.

MindSpore is designed to support transition between data parallelism and model parallelism during model training. To this end, tensor redistribution (TR) is introduced into the search of parallelism strategies. As shown in Figure 3, TR transforms the layout of an output tensor among devices before the tensor is fed into the subsequent operator. On this basis, MindSpore defines the corresponding backward operator to realize the derivation of the communication operator. Doing so enables the AD procedure to differentiate the entire forward graph in one hit. In addition, MindSpore builds a cost model to select a suitable strategy that achieves a balance between computation and communication overheads. Two techniques are proposed to quickly find such a strategy for complex and large graphs: One is an algorithm that supports multi-graph operations and transforms the original graph into a linear one. The other is a strategy sparsification mechanism that effectively shrinks the search space while also guaranteeing the accuracy of the returned solution.

MindSpore supports flexible user-defined high-level strategy configuration, also referred to as semi-auto parallelism. Furthermore, MindSpore can find an effective parallelism strategy without manual intervention, avoiding repetitive strategy configurations every time a new model is trained.

3.1.3 Dynamic Graph

Static graphs generally offer better runtime performance because the compiler is fed with global information. Conversely, dynamic graphs offer greater ease of use, allowing users to build and edit models easily. To support both static and dynamic graphs, most frameworks need to

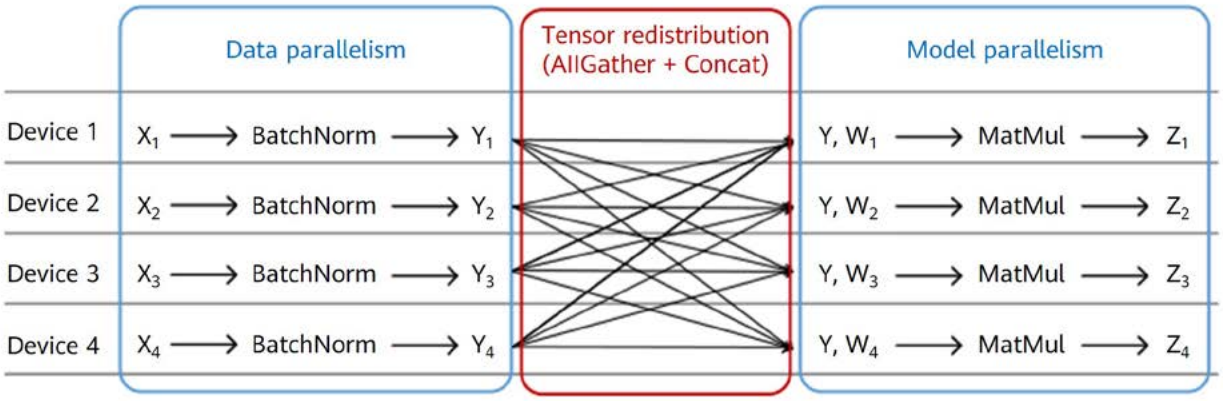


Figure 3 Transforming data parallelism to model parallelism

maintain two AD mechanisms: tape-based AD and graph-based AD. This introduces maintenance costs and complexity in switching between the two modes.

MindSpore takes a different approach and implements a unified SCT-based AD mechanism to support both static and dynamic graphs, requiring only one line of code to switch from one mode to the other. In static graph mode, the neural network is compiled into a full computational graph before it is executed. This mode introduces graph optimizations in the process to improve runtime performance and facilitates large-scale deployment and cross-platform execution. In dynamic graph mode, operators in the neural network are delivered and executed one by one, making it easy to build and debug neural network models. In addition, MindSpore supports a staging mechanism that facilitates hybrid execution. To compile a function in static graph mode, we simply add the **ms_function** decorator in front of the function.

3.1.4 Second-Order Optimization

Optimizers are the core of backpropagation algorithms and play an important role in deep learning. Common optimization algorithms are classified into first-order and second-order optimization algorithms. Among first-order optimization algorithms, classical gradient descent (GD) is most widely used in machine learning. GD brings many algorithm variants, such as Momentum [11], AdaGrad [12], and Adam [13]. These improved variants are easier to tune and use, because they can adaptively update the step size by using historical information about stochastic gradients.

Given that neural networks have highly non-convex loss functions with unbalanced surface curvature, using more information in parameter updates helps speed up convergence. In this regard, second-order optimization algorithms leverage the second-order derivative of the target function to correct the curvature and accelerate first-order descent. Faster convergence enables these algorithms to better approximate to the optimal value and yield a geometric descent path that is closer to the optimal real-world one. Compared with first-order algorithms, second-order optimization algorithms update the weight based on the inverse matrix of the second-order information matrix. Typical second-order optimization algorithms are Newton method and natural gradient method, whose corresponding second-order information matrices are the Hessian matrix and Fisher matrix, respectively.

Although second-order optimization algorithms feature faster convergence, the time complexity involved in computing the inverse of a second-order matrix is extremely high. Because a deep learning model can have millions of parameters, it is extremely complex to compute the inverse of second-order matrices. Therein lies the crux of the matter: how to reduce the computational complexity of second-order matrix inversion.

To address this issue, MindSpore provides a novel algorithm: trace-based hardware-driven layer-oriented natural gradient descent computation (or THOR for short) [14]. This algorithm enhances second-order algorithms based on the natural gradient method in the following three aspects: (1) Matrix update frequency. The Frobenius norm (F norm) of the Fisher matrix changes acutely in the early stage and gradually becomes stable in the later stage. Therefore, it is assumed that the F norm is a Markov process and can converge to a steady-state distribution. Gradually increasing the update frequency of the Fisher matrix is sufficient to shorten the training time without compromising the convergence speed. (2) Layer-wise update. Fisher matrices are decoupled layer-wise, and some of these layers reach

the steady state faster than others do. This inspired us to finely adjust the update frequency of each layer based on the trace change of the second-order information matrix. (3) Hardware-aware matrix splitting. THOR assumes that Fisher matrices are decoupled layer by layer and that the input and output blocks at each layer are independent of each other. Based on this assumption, THOR further splits second-order matrices to improve computational efficiency.

3.2 Graph Kernel Fusion

As deep neural networks (like GPT-3 [15]) become larger, the runtime performance and efficiency of those networks become limited by our ability to optimize them. Existing compiler optimization techniques such as Accelerated Linear Algebra (XLA) [16] and Tensor Virtual Machine (TVM) [17] have achieved excellent results on different AI frameworks.

It is demonstrated that optimization methods that consider both graph and operator layers based on automatic operator compilation can effectively improve model performance and runtime efficiency. MindSpore proposes the graph kernel fusion technique, where transforming the graph facilitates the generation of efficient operators and, in turn, the operator generation rules guide how the graph is transformed. Graph kernel fusion optimizes graph performance from three dimensions: load/store fusion, parallel fusion, and cross-boundary kernel optimization.

Load/Store Fusion

A computational graph consists of operators connected by data or control links. On the one hand, using graphs maintains the computing integrity and independence of each operator and operator-specific optimization, but also eliminates the possibility of operator optimization at higher levels. On the other hand, when operators have large inputs/outputs, data loads and stores can significantly prolong model runtime and become a performance bottleneck. In the load/store fusion process, upstream and downstream operators are fused based on their load/store patterns and special processing is applied on poor composite operators, thereby effectively reducing memory footprint with improved computing density.

To maximize the computing logic that an operator boundary can express while also minimizing the total operator boundaries, graph kernel fusion attempts to fuse as many operators as possible and then disassembles and reallocates the generated composite boundaries. Redesigning operator boundaries offers more optimization opportunities at acceptable load/store loss. MindSpore's Auto Kernel Generator (AKG) implements polyhedral-based auto scheduling, covering auto vectorization, auto segmentation, thread/block mapping, dependency analysis, and data transfer. A range of optimizations are also introduced for the back-end, including Tensor Core enabling, dual buffers, memory expansion, and insertion of sync instructions. Thanks to its powerful analysis capability, the AKG can generate high-quality executable operators based on operator descriptions. To support graph kernel fusion, the AKG enables auto generation of operators with high performance.

Graph layer transformation is another important factor for operator performance. The AKG's experience in generating operators and backend optimization offers insights for the redesign of operator boundaries. To reduce loads/stores and optimize computations related to operator boundaries, graph kernel fusion aggregates and then disassemble operators on the graph, thereby realizing computing optimization beyond operator boundaries. This steps over the optimization barriers involved in singular computing logic, without blindly expanding the fusion range — only boundaries that bring benefits (as per the evaluation result) are generated. As such, load/store is a means of performance improvement. With the capability of generating operators automatically, the AKG frees graph kernel fusion from the restrictions of one-to-one manual fusion and enables flexible and effective operator fusion possibilities. Load/Store fusion significantly improves the computing density and effectiveness for operators on the graph, and reduces inefficient loads/stores across operator boundaries, thereby improving the overall execution performance of the graph.

Parallel Fusion

Proper execution of a computational graph relies on the defined execution sequence of computing tasks. Such a sequence typically includes implicit execution dependencies. When many fine-grained computing tasks are loaded to hardware for execution, the hardware remains idle most of the time, resulting in a waste of compute resources. Parallel fusion is proposed to analyze the dependencies between operators on a graph before fusing operators with parallel properties and low hardware usage. For a generated parallel operator, each part is independently scheduled and allocated with independent compute resources. By fusing operators with low hardware usage into composite ones, parallel fusion improves hardware usage, avoids the waste

of compute resources, and improves execution performance of the graph.

Cross-Boundary Kernel Optimization

The computing logic of coarsely aggregated operators is not optimal and requires optimization before they are further disassembled. Prior to aggregation, however, a number of optimization barriers exist. For example, an operator can only be completely referenced due to operator boundary. Referencing a composite operator to use its partial output result will inevitably introduce redundant operations into the network. If two composite operators share similar operations, it is impossible to split or combine the operations before the operators are aggregated. Fortunately, such correlations become explicit and optimization becomes possible when the exposed computing logic is aggregated. With cross-boundary kernel optimization, we can eliminate common subexpressions and dead code in the aggregated operation blocks to remove unwanted operations with improved computing efficiency.

3.3 MindData

MindData is a separate asynchronous data processing pipeline that prepares input tensors for the model. Data is organized into rows and columns, and each column is identified with a name and can be accessed independently. The pipeline always starts with a source dataset operator, which reads data from disks (e.g., MindDataset) and provides flags for selecting between the shuffling and sharding strategies. An iterator for Python access and device queues that directly send data to accelerators are provided for access to data in the pipeline.

Data processing tasks are intrinsically pipelined and parallelized. The pipelines run asynchronously by default. Users can insert sync points into graphs to support real-time feedback loops for pipeline operators. To optimize performance, users only need to configure default parameters, eliminating the need for manual tuning. In the future, pipelines will support dynamic tuning to fully utilize all available resources, including hardware accelerators for image processing or available memory for caching.

To enable quick migration to MindSpore, users can port existing Python data as custom operators and pass existing Python dataset classes as arguments to MindData. Workloads are emerging with new requirements for data processing to support greater flexibility. MindData allows users to adjust parameters (e.g., the mini-batch size) using user-defined functions or schedules. Users can also perform custom transforms on an entire mini-batch to support manipulations at the mini-batch level, including image size alterations and multi-row operations such as image mix-up. For greater augmentation diversity, the augmentation of each sample can be randomly chosen from sets of transforms. External search is supported for selecting among transform candidates. It is demonstrated that randomly selecting transforms from a wide variety of candidates produces results comparable to those obtained without the additional search time. In addition, feedback from loss or other metrics collected during training can be passed back into the dataset to perform dynamic adjustments in MindData.

Before training data is read into MindDataset, it is stored and indexed in MindRecord according to different types and pages in a lightweight and efficient manner. MindDataset supports fast fetch of important metadata (e.g., dataset size or data layout) from the dataset to improve performance or simplify user access. In addition to supporting sequential I/O of small data blocks, MindRecord also supports efficient random row-access and push-down filtering as per use-case requirements. As new use cases emerge, further optimized functions will be pushed down into the dataset.

3.4 MindInsight

MindInsight is MindSpore's visualized debugging and optimization tool. It displays the training process, optimizes model performance, debugs accuracy errors, and explains inference results in a visualized manner. It also provides a command line interface for developers to search for hyperparameters and migrate models. MindInsight allows developers to better observe and understand the training process with improved model optimization efficiency and developer experience, making it easy to obtain satisfactory model accuracy and performance.

MindInsight uses the summary file generated during model training as the input. By performing the file parsing, information extraction, data caching, and chart plotting steps, MindInsight transforms binary training information into intuitive charts displayed as web pages. It also visualizes the training process via the training dashboard, providing a training overview on one page. Modules on the training dashboard include training scalar information, parameter distribution, computational graph, data graph,

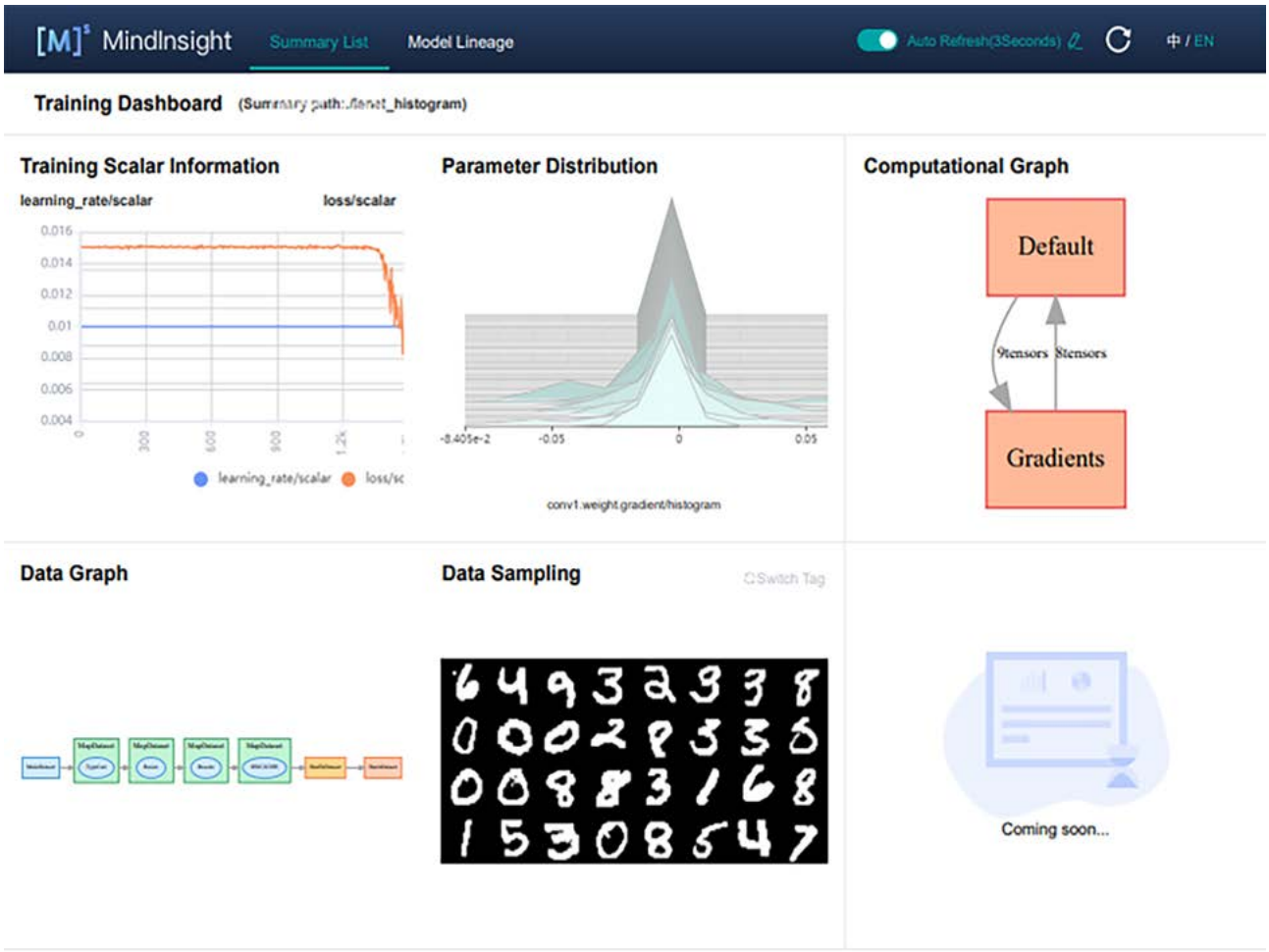


Figure 4 MindInsight training dashboard

and data sampling. Figure 4 shows an example of the training dashboard.

MindInsight also supports lineage visualization by integrating lineage information of multiple training runs into tables and charts, where users can easily find the optimal data processing pipeline and hyperparameter settings. Lineage visualization covers model and data lineage visualization. Model lineage records key parameter information about model training, such as the loss functions, optimizers, number of epochs, and accuracy. Furthermore, MindInsight displays hyperparameters that are trained multiple times and the corresponding metrics, helping users select the optimal hyperparameter.

Another important feature of MindInsight is the profiler. It collects statistics such as execution time and memory footprint of each operator. MindInsight then sorts and analyzes the profile data and presents the analysis results from step trace, operator performance, timeline, and MindData profiling dimensions, providing valuable information for optimizing network performance.

During the training of neural networks, numerical errors such as infinity can prevent training from converging. Because computations are performed in a blackbox manner in the graph execution mode, locating these errors is difficult. The MindInsight debugger allows users to view the internal structures of a graph and inputs/outputs of each node in graph execution mode during training. For example, users can view the value of a tensor, map a node on the graph to Python code, and set a conditional breakpoint for a group of nodes in order to monitor the compute results of the nodes in real time.

3.5 MindArmour

Adversarial attacks [19, 20] have become an increasingly prevalent threat to the security of machine learning models. An attacker can compromise machine learning models by adding small perturbations that are difficult to spot to the original sample [21, 22]. To defend against adversarial attacks, MindArmour offers three main modules: attack

(adversarial example generation), defense (adversarial example detection and adversarial training), and evaluation (model robustness evaluation and visualization).

Taking a model and data as input, the attack module generates corresponding adversarial examples in both blackbox and whitebox attack scenarios using easy-to-use APIs. These examples are then fed into the defense module to improve the generalization and robustness of the model. The defense module also implements multiple detection algorithms, which can distinguish between adversarial examples and benign ones based on either malicious content or attacking behaviors. The evaluation module provides multiple evaluation metrics, enabling developers to easily evaluate and visualize the robustness of their models.

Privacy preservation is a major topic in AI applications. MindArmour provides effective solutions to the main privacy-preserving issues. In order to provide differential privacy guarantee to a trained model (which might leak sensitive information about the training dataset [23, 24]), MindArmour implements a series of differential privacy optimizers that automatically add noise to the gradients generated during backpropagation. Specifically, the optimizers adaptively add noise in the training process, achieving better model utility with the same differential privacy budget. Users can use these differential privacy optimizers in the same way as they would use normal ones, and they can dynamically monitor privacy budget consumption during training using the monitoring module.

3.6 Device-Cloud Collaboration

MindSpore aims to build an AI framework that covers all scenarios, from the device to the cloud. To this end, MindSpore implements a wealth of "device-cloud" collaboration capabilities, including model optimization, on-device inference and training, and federated learning, as illustrated in Figure 5.

Model Generation and Optimization Toolkit

To help users deploy models on mobile and edge devices that have limited resources (such as power and memory), MindSpore implements a collection of optimization techniques (as shown on the left side of Figure 5). Model adaptive generation is a method that adaptively generates models according to specified device, latency, accuracy, and size settings using neural architecture search (NAS) technology [25]. A quantization strategy is used to approximately represent 32-bit floating-point data by using a data type with fewer bits. Both post-training quantization and quantization-aware training are supported in MindSpore.

On-Device Training and Federated Learning

Although deep learning models trained on large datasets can be generic to some extent, they do not apply to personalized user data or tasks in some scenarios.

MindSpore's on-device training solution allows users to train personalized models or fine-tune existing ones on their devices without facing data privacy, bandwidth limitation, and Internet connection issues. In the near future, various on-device training strategies — including training from

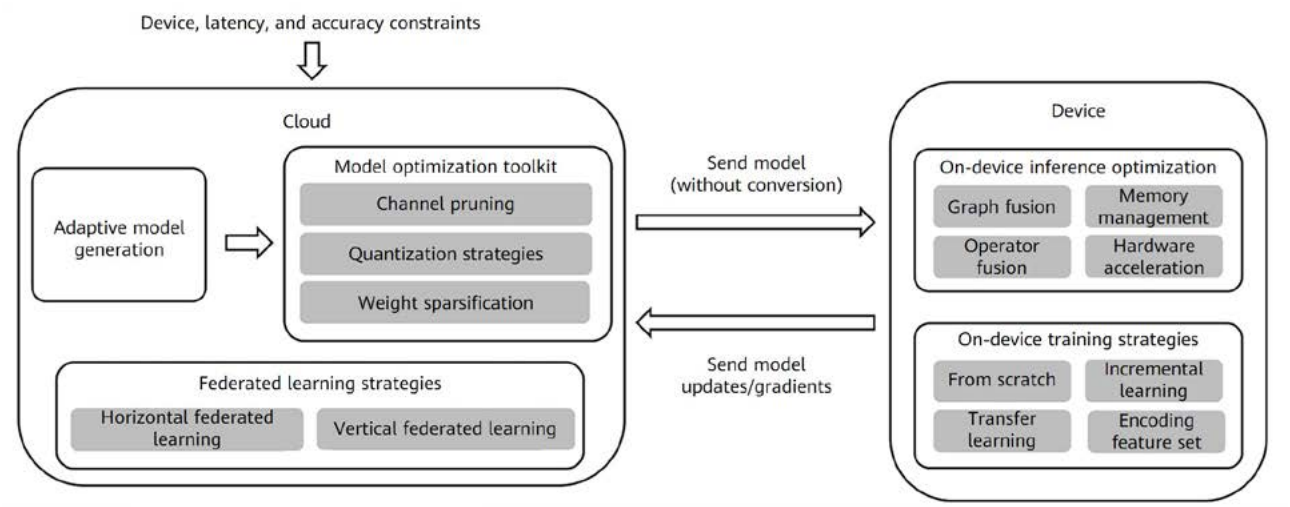


Figure 5 MindSpore device-cloud collaboration architecture

scratch, transfer learning, and incremental learning — will be brought to MindSpore. MindSpore also supports federated learning to share data by sending model updates/gradients to the cloud. With federated learning, models can learn a greater volume of general knowledge.

Deployment on Mobile and Edge Devices

MindSpore provides a lightweight compute engine for executing models efficiently on devices. Typically, model conversion is necessary before pre-trained models can be deployed to the device side, potentially leading to unexplainable performance and accuracy degradation. In MindSpore, the on-device inference schema is compatible with on-cloud training, eliminating the need for such conversion and avoiding the potential performance degradation. Furthermore, MindSpore features a variety of built-in automatic optimization methods targeting devices, such as graph/operator fusion, fine-grained memory management, and hardware acceleration, as shown on the right side of Figure 5.

3.7 MindSpore Serving

MindSpore Serving is a lightweight high-performance inference module designed to help MindSpore users efficiently deploy their online inference services in production environments. It allows users to create inference services based on pre-trained models exported from MindSpore.

MindSpore Serving helps users complete the following tasks: loading model files to generate an inference engine and providing the inference function; exchanging gRPC or RESTful messages of prediction requests and results; prediction interface calls, prediction processing, and returning the prediction results; managing the lifetime of models; managing the lifetime of services; and managing multiple models and versions.

3.8 GNN Framework

GNNs have attracted significant interest from both academia and industry. Application scenarios include drug identification and discovery, recommendations, traffic flow prediction, and chip design. Like CNNs, GNNs are also evolving quickly. However, existing GNN frameworks (e.g., DGL [26], PyG [27], PGL [28], and GraphLearn [29]) fail to provide a simple and easy-to-use programming model that fits GNN algorithms while also providing satisfactory training speed. The MindSpore developers worked with the Chinese University of Hong Kong to propose an easier and faster GNN framework. As shown in Figure 6, the framework is built on a node-centric programming model and a wealth of algorithms for deep graph learning and compilation optimization.

In GNN tasks, users need to transfer and aggregate information on given graphs (consisting of edges and nodes). This approach differs significantly from that used in CV tasks. Existing frameworks fail to support intuitive expressions like Python syntax for such operations. Users rely on the methods (e.g., message-passing) provided by GNN systems to realize information aggregation. Such a tensor-centric programming model creates a steep learning curve for users, compared with intuitive Python implementations.

MindSpore implements a node-centric programming model in its GNN framework. This model fits users' thinking and usage habits as closely as possible, enabling users to build a GNN model as easily as writing a common

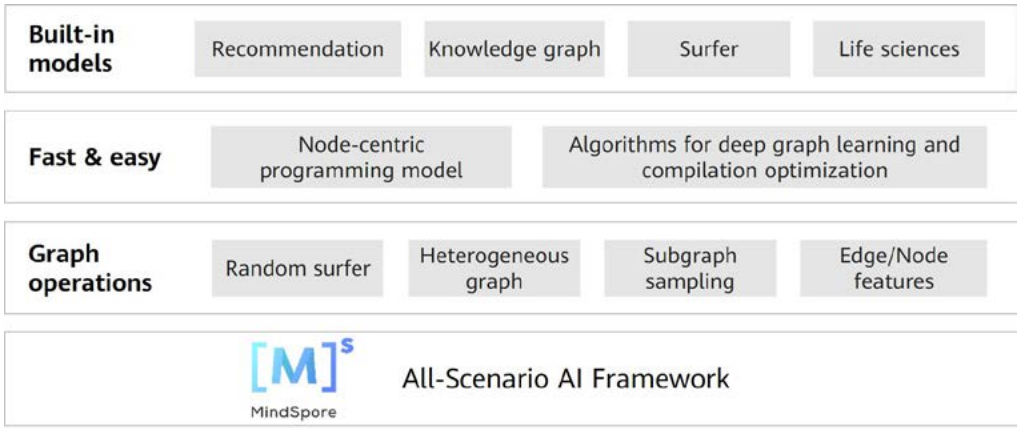


Figure 6 MindSpore GNN architecture

Python program. Furthermore, this model significantly reduces the threshold for developing new models and facilitates model implementation and iteration. It provides equivalent expression capabilities as the message-passing method and is compatible with all existing GNN models. Based on this programming model, MindSpore quickly reproduces the convolution library and model library of the Deep Graph Library (DGL) [26], the most popular GNN framework. Currently, more than 10 classic models are supported, covering homogeneous and heterogeneous models, recommendation models, knowledge graphs, and life sciences. This fully proves the utility and expression capability of this programming model.

The node-centric programming model faces the following challenges: First, adaptation to existing MindSpore code for seamless connection between the two programming models. Second, compatibility with MindSpore's existing features (e.g., auto parallelism, easy switching between dynamic and static graph modes, and compilation of computational graphs) and new features to be launched. Third, minimizing intrusions to the existing MindSpore framework. To address these three challenges, an innovative source code transformation method is proposed in the MindSpore GNN framework to convert user code into a format natively supported by MindSpore, allowing for data exchange and connection in tensor format between different programming models. The result of source code transformation is MindSpore code, which ensures full compatibility with MindSpore features without requiring any alterations to the existing framework. The GNN framework also provides a convenient tool for comparing the generated code against the source code line by line.

Typical GNN tasks are memory-intensive, and the core overhead of these tasks lies in frequent tensor movements between the RAM, cache, and processor core. Conventional methods use operator fusion to reduce memory movements. In existing GNN frameworks, available operator fusion optimization schemes are mainly a point-by-point optimization approach, where users have to manually define common GNN operator combinations and write forward and backward operator fusion code. This approach is not generic, requiring developers to manually define and optimize GNN operator fusion. Furthermore, the operator fusion opportunities are not fully realized, resulting in only average performance.

Conventional GNN tasks run in Gather-Injective (+/-/*// and activation)-Scatter (GIS) mode when homogeneous and heterogeneous graphs are converted into MindSpore computations through the scatter/gather process. To enhance performance and scalability, MindSpore proposes a new GIS mode for GNN operator fusion optimization based on the graph kernel fusion and auto operator compilation techniques built into the framework. This new GIS mode can fuse one or more Gather, Injective, and Scatter operators into one large operator. After operator fusion, the polyhedral-based technology is leveraged to automatically optimize and generate operators at the AKG operator compilation layer. For large data volumes, some operators in the resulting computational graph have ultra-large output results. Given that the bottleneck of fused operators lies in memory footprint rather than computation, we can improve the execution speed by introducing recomputation, thereby reducing the RAM usage. With reduced RAM instantiation of tensors, MindSpore's GNN framework outperforms DGL (the most popular existing framework) by an average of three to four times. This fusion method applies to the optimization of not only existing operator fusion in the existing framework, but also many more new operator combinations.

4 MindSpore Ecosystem

Since becoming open source a year ago, MindSpore has grown into the most active open-source community in China and there have been 12 versions released to date. MindSpore ecosystem is expanding swiftly across universities, competitions, institutes, and enterprises with rapid iterations of technological innovation. Put simply, MindSpore has created a virtuous "industry-education-research-competition" cycle in the AI industry: The industry attracts and retains talent at the forefront of research and innovation, and this in turn drives the advancement of the industry.

Universities are the base for talent cultivation and the future of AI talent development. They are also the starting point of the MindSpore ecosystem (see Figure 7). China's Ministry of Education has initiated the "Emerging Engineering Education" plan, aiming to cultivate AI talent and build a world-class science center and talent resource pool. Against this backdrop, the MindSpore ecosystem has sought close cooperation with universities in order to streamline the "industry-education-research-competition" cycle, build an industry-driven AI knowledge system, and prime the talent pipeline for continuous AI innovation. In addition to building influence at the technological frontier through close cooperation with academia, MindSpore has

been promoting industrial implementations (healthcare, transportation, finance, Internet, telecom, manufacturing, energy, etc.) while also improving AI model R&D efficiency and lowering the threshold of AI application implementation for customers from mature industries. MindSpore takes its root in AI infrastructure, develops a high-level application ecosystem, consolidates low-level hardware capabilities, and expands its reach internationally.

The following discusses our experience in MindSpore ecosystem building in academia and business perspectives as well as in community operations.

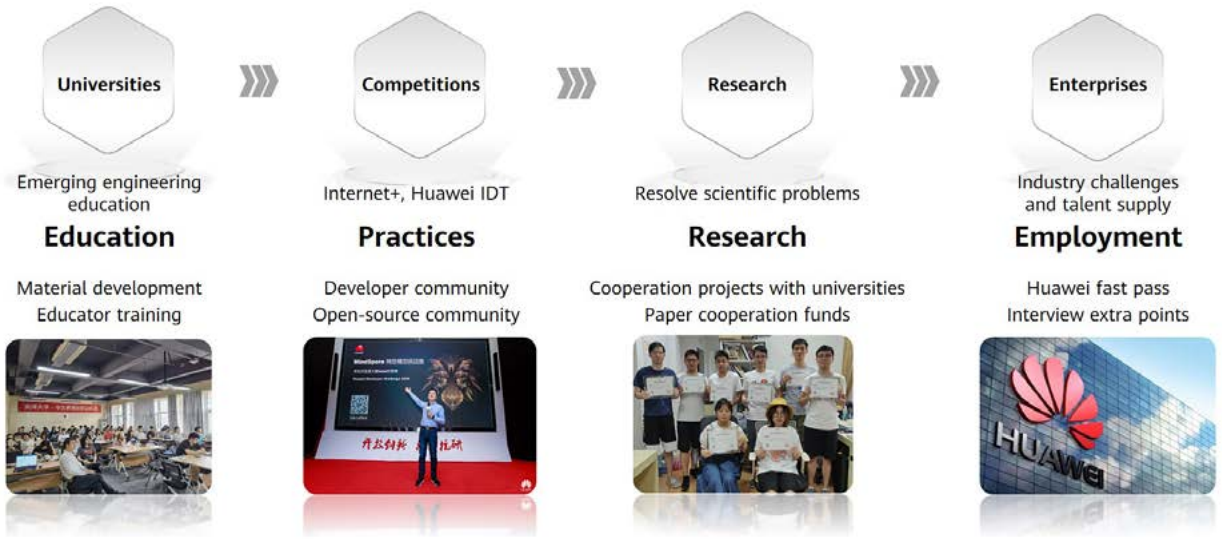


Figure 7 Virtuous "industry-education-research-competition" cycle of MindSpore ecosystem

Figure 8 MindSpore-based education in universities

4.1 Ecosystem Building in Academia

MindSpore cooperation with universities begins with three programs: Emerging Engineering Education, Intelligence Foundation, and Developer Program. MindSpore offers curriculum development and educator training for university teachers to prepare them to teach AI-related subjects based on MindSpore. After learning about MindSpore, junior undergraduates are encouraged to take part in developer activities and competitions in the community, Internet+ competitions recognized by the Ministry of Education, and the MindSpore racetrack of Huawei ICT Competition. For students interested in academic research, MindSpore and the Chinese Association for Artificial Intelligence (CAAI) have initiated an academic reward fund to provide them with technical, capital, hardware, and compute power support for scientific research based on MindSpore.

MindSpore Opening University Courses in This Autumn Term

MindSpore has completed 10 educator training sessions in eight cities (including Chengdu, Wuhan, and Shanghai) and started 416 courses in 101 of the top universities in China (including Tsinghua University and Peking University). It has reached over 1000 teachers and 35,000 students, covering important AI and computer science topics such as ML/DL, CV, and NLP. In the 2021 autumn term, 159 new courses led by more than 200 teachers are rolling out for 10,000 students. More than 10 MindSpore Study Group (MSG) sessions have been held, attracting more than 5000 teachers and students (see Figure 8). MindSpore has promoted AI education in universities and achieved substantial breakthroughs in ecosystem building in academia.

MindSpore in Internet+ and Huawei ICT Competitions

Hands-on practice is proven to be one of the most effective methods for enhancing learning. MindSpore has created questions in major AI competitions, attracting tens of thousands of students to learn and practice MindSpore and take part in the open-source community (see Figure 9).

1. In the industry racetrack of the Internet+ Competition, 170 teams signed up for MindSpore-based contests. Among them, 13 teams won provincial gold medals and were shortlisted as the national top 300, and four teams were selected as the national 50 finalists.

2. More than 3000 teams have signed up for the NSFOCUS Cup. In the current golden stage, eight teams will be selected as MindSpore representatives to compete for the annual championship at the China Computer Federation (CCF) Conference in Xi'an from November 19 to 20.

3. The Open Source Promotion Plan was executed successfully. More than 200 teams applied for MindSpore projects. As the projects successfully closed, the project code was contributed to the community and 30 developers stood out as core community contributors.

4. There are a number of other ongoing MindSpore racetracks in the Huawei ICT Competition, CCF Big Data & Computing Intelligence Contest (CCF BDCI), AI+remote sensing and AI+manufacturing competitions founded by China Academy of Information and Communications Technology (CAICT), and more.

Figure 9 MindSpore in Internet+ competitions

40+ Joint Research Projects Published 170+ Papers

MindSpore is driving the growth of the AI ecosystem through research cooperation and technological innovation. MindSpore has cooperated with more than 20 top universities and research institutes around the world to enhance its competitiveness in fields such as AI+scientific computing, trillion-parameter super large models, and AI security. Furthermore, it has published a series of research achievements including remote sensing models, mobile electromagnetic simulation, and multimodal foundation models.

MindSpore and CAAI jointly initiated an academic reward fund that offers capital, compute power, and technical support for university and institute AI researchers. This fund aims to promote MindSpore-based academic application and paper publication in major national and international conferences and periodicals, while also bolstering global influence of China's AI research with incentives on originality.

To date, this fund has established partnerships with more than 45 university educators on many popular AI fields (CV, NLP, scientific computing, biology, autonomous driving, etc.) and supported the publication of over 170 MindSpore-based research papers at major conferences.

4.2 Ecosystem Building in Business

This year has witnessed the burgeoning rollout of AI computing centers in Shenzhen, Wuhan, Xi'an, Chengdu, and Nanjing. Among them, the Shenzhen and Wuhan centers have already been put into operation. These AI computing centers provide inclusive computing power to local universities and enterprises, support national and international efforts in joint innovation based on MindSpore, promote the domestic AI developer ecosystem, and empower industrial practices.

PCL-L: First Open-Source Chinese Pre-trained Language Model with 200 Billion Parameters

At Huawei China Ecological Conference, Peng Cheng Laboratory (PCL) unveiled PCL-L, the industry's first open-source Chinese-language pre-trained foundation model with as many as 200 billion parameters. Based on Peng Cheng Cloud Brain II, this foundation model employs the auto hybrid parallelism approach provided by the MindSpore framework, and implements 200-billion-parameter distributed training on a 2048-card cluster. It is the first time for domestic full-stack AI infrastructure to support language model training on such a large scale. This demonstrates the breakthroughs of domestic exascale supercomputing platforms in core capabilities such as software and hardware co-optimization and large-scale distributed parallel training.

Large-scale distributed training has been a major obstacle to the development of foundation models. MindSpore tackles this obstacle by offering a full collection of hybrid parallelism choices: data parallelism, operator-level model parallelism, pipeline model parallelism, optimizer model parallelism, and recomputation. MindSpore introduces the combined use of multi-dimensional parallelism techniques in the graph compilation phase.

PCL-L enables industry-leading performance in a wide range of applications, especially in text generation fields such as Q&A, knowledge retrieval, knowledge inference, and reading comprehension. Against state-of-the-art (SOTA) benchmarks, PCL-L delivers the best performance indicators in 16 downstream tasks. Specifically, PCL-L achieves top marks in 11 zero-shot learning tasks, 12 one-shot learning tasks, and 13 few-shot learning tasks.

MindSpore and Wuhan University Create the First Remote Sensing–Dedicated Framework

Remote sensing technology is widely used in land resource planning, natural environment monitoring, and similar fields. Due to the nature of this technology, the interpretation of remote sensing images is more complex than that of general image recognition. As such, most remote sensing mapping tasks currently rely on manual interpretation. The application of AI and deep learning technologies in solving remote sensing problems requires a sufficiently large remote sensing image sample library, a dedicated deep learning framework, and a model specifically designed for remote sensing. As a tier one university teaching remote sensing majors, Wuhan University collaborated with Huawei Ascend and MindSpore to build the world's first deep learning framework LuojiaNet and sample library LuojiaSet for remote sensing usage (see Figure 10).

MindSpore parallelizes the processing and segmentation of ultra-high-resolution images to multiple devices and cards, solving the boundary computation problem of image segmentation. Built based on MindSpore, the LuojiaNet framework is dedicated for promoting remote sensing research and applications in China. It supports remote sensing images with a resolution of up to 30000 x 30000

pixels and 256 channels and provides features such as abstract remote sensing knowledge graph. On top of the remote sensing sample library and the dedicated framework and model, Wuhan is cultivating its remote sensing industry by supporting startups such as Optics Valley Information and Leador Spatial Information.

MindSpore and Chinese Academy of Sciences Build a Generic AI Platform and Develop the First Three-Model Foundation Model

Pre-trained models like GPT and BERT have achieved major success. Such models learn different tasks without supervision and can be quickly migrated to different fields. Multimodal pre-trained models are considered the way forward for moving from weak AI to general AI. In recent years, audio and video data has grown at an unprecedented rate and contributed to over 80% of all data. However, the pre-trained models currently available are mainly designed for text workloads. The massive volumes of speech, image, and video data are not utilized and learned. In fact, human information acquisition, environment perception, and knowledge learning and expression are implemented in multimodal mode.

The Institute of Automation of the Chinese Academy of Sciences proposed a visual-text-speech three-modal pre-trained model and successfully built the world's first three-modal pre-trained foundation model "Zidong.Taichu" based on the MindSpore framework (see Figure 11). The model features cross-modal understanding and generation. Specifically, it can generate images from speech and vice

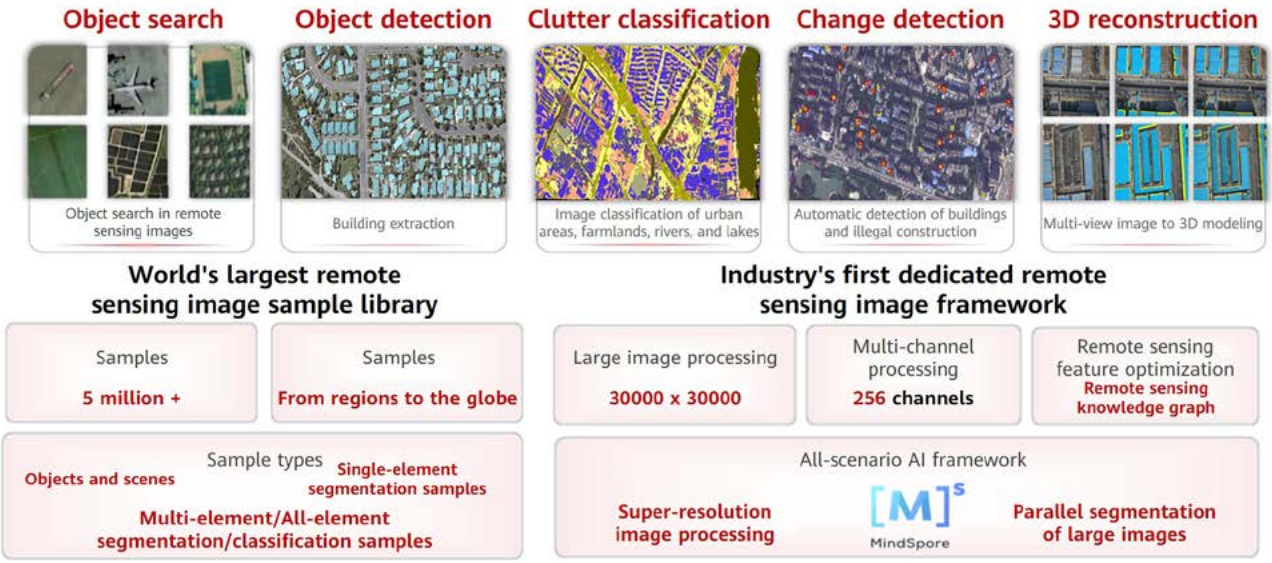


Figure 10 Industry's first sample library and dedicated framework for remote sensing

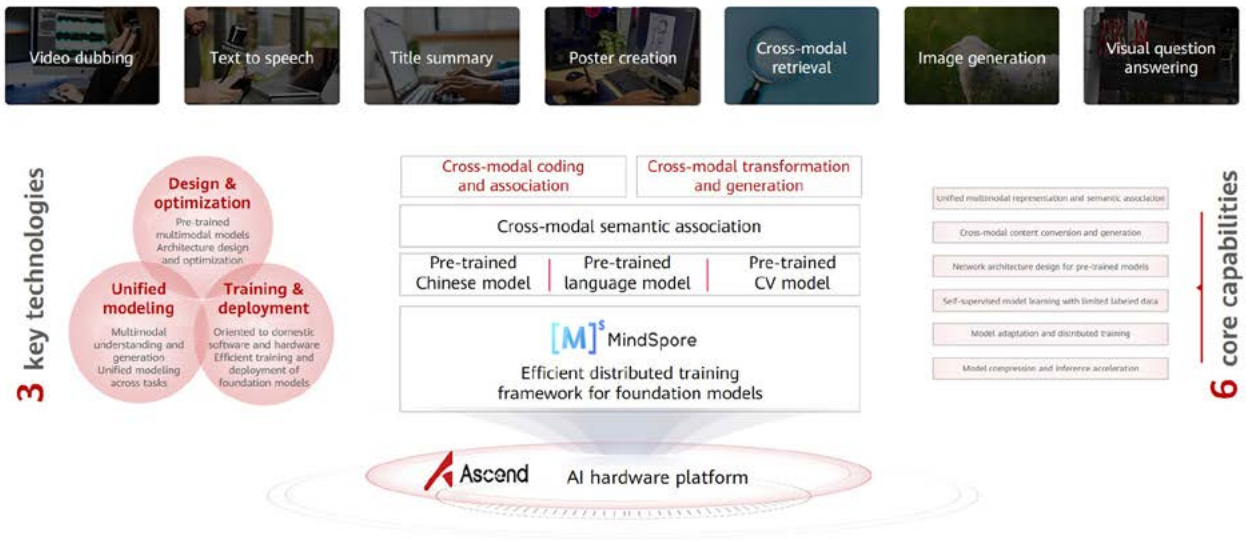


Figure 11 Zidong.Taichu — industry's first three-modal foundation model

versa, and it can realize unified semantic representations of images, text, and speech. This model is pre-trained on 100 TB of three-modal data on a 100-card Ascend cluster in distributed mode. Distributed training of this model is significantly accelerated with lower optimization costs thanks to the multi-dimensional and full-type distributed parallelism strategies provided by MindSpore, from data parallelism, operator-level model parallelism, pipeline model parallelism, optimizer model parallelism, and heterogeneous parallelism, to recomputation and memory reuse.

The Zidong.Taichu foundation model achieves superior performance with efficient collaboration between text, visual, and voice models. It outperforms SOTA benchmarks in visual-text cross-modal understanding and generation, and it works efficiently in downstream tasks like cross-modal detection, visual Q&A, and semantic description. Zidong.Taichu remarkably won the video understanding and video scene parsing challenges in ACM Multimedia 2021 and ICCV 2021. The performance of Zidong.Taichu has attracted attention from enterprises like SAIC Motor, Weiqiao Pioneering, and iQIYI for cooperation projects in fields such as smart driving, industrial quality inspection, and film production.

MindSpore and Xi'an Universities Build Industry's First Chinese Speech Foundation Model and Intelligent Radar Remote Sensing Applications

In September 2021, Xi'an Future AI Computing Center commenced operations. MindSpore, Xidian University, Northwestern Polytechnical University, and Shaanxi Normal University reached a cooperation agreement on civilian application innovation, centering on building intelligent remote sensing and Chinese and multilingual speech foundation models.

Oriented to radar remote sensing images, Xidian's intelligent remote sensing technology enables all-day observation and analysis with immunity to weather interference, even in extreme weather conditions like earthquakes, rainstorms, and typhoons. This project mainly targets surface features extraction, registration, and change detection (see Figure 12). This is especially useful in marine monitoring, mountain change monitoring, urban development, facility expansion, and vegetation changes, and particularly in post-disaster assessment and rapid disaster relief.

Speech is the most natural form of communication. Existing speech models are highly dependent on labeled data, but labeling speech data costs much more than labeling image and text data. MindSpore works with Northwestern Polytechnical University to develop large-scale speech pre-trained models that can achieve system robustness with only a small amount of labeled data. The project aims to improve the comprehensive speech processing capabilities in different fields, dialects, and languages, and provide rich AI applications including self-service IVR, speech synthesis, spoken language understanding, speech-to-speech translation, and voiceprint recognition. The project has built four foundation models: (1) The world's first Chinese speech pre-trained foundation model. (2) The world's first speech pre-trained model for Chinese dialects, improving the

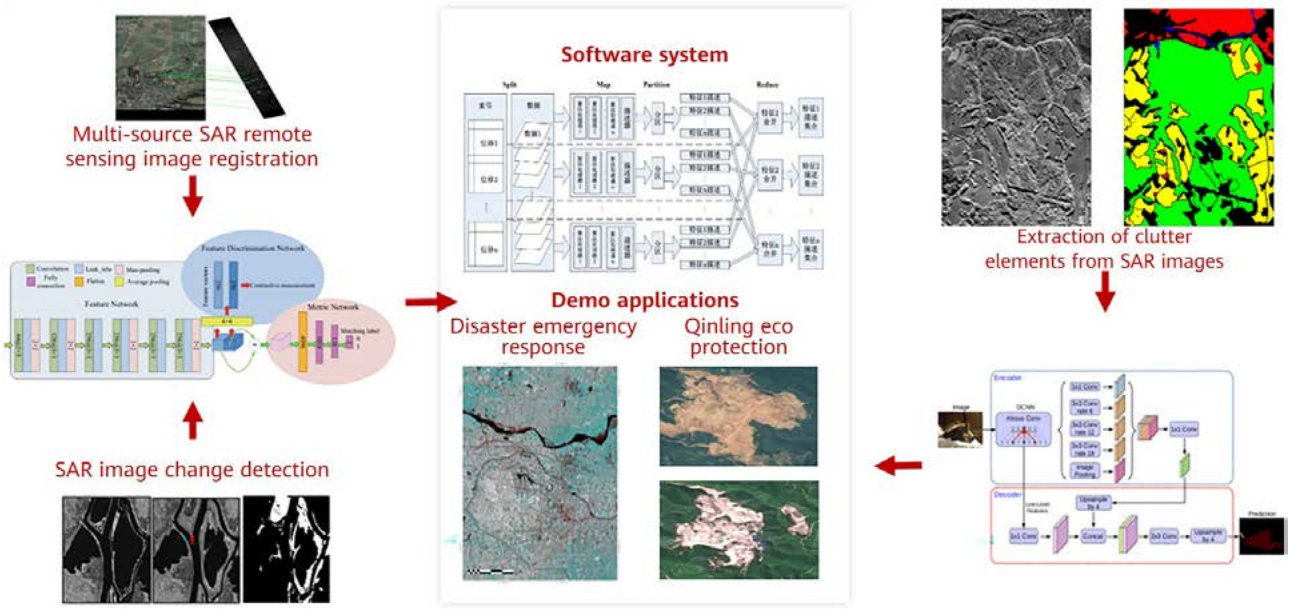


Figure 12 Intelligent radar image pre-trained model

comprehensive processing capability of accents and dialects; (3) A multi-language speech pre-trained model, improving the multilingual speech processing capability. (4) A pan-audio foundation model for perceiving and understanding voices.

Enabling Industry Upgrade by Embracing Open Source and Openness

MindSpore has set up industry benchmarks in eight industrial cases (including healthcare, transportation, finance, manufacturing, and energy) and worked with industry partners to develop solutions for smart healthcare, smart finance, smart manufacturing, etc. More than 60 industry partners have passed the MindSpore certification, and the number is expected to reach 100 by the end of this year.

The MindSpore-based smart manufacturing solution (see Figure 13) has been quickly replicated in the production lines of PowerLeader, Yangtze River Computing, Midea Hefei, and Haier. This solution has improved detection accuracy for the assembly workshop of Yangtze River Computing by 10% and enabled fast upgrade of production line algorithms within 2 hours. In terms of smart retail, Wesine launched a MindSpore-based intelligent AI loss prevention system to improve retail efficiency.

At the ecosystem partner workshop sponsored by MindSpore in Wuhan, Dofront Technology, Optics Valley Information, Leador Spatial Information, Douyu Network, Industrial Control Research Institute, Lottop, Wesine, Nasi Technology, JIMU Intelligent, and Welltrans O&E exchanged ideas on development trends and cooperation opportunities.

Pair Enables Intelligent Image Annotation, Freeing Doctors from Manual Annotation

Pair is an easy-to-use and intelligent portfolio of medical image annotation software (see Figure 14). It is developed by Professor Ni Dong and Doctor Yang Xin of the Ultrasonic Image Computing Lab of Shenzhen University.

Refined segmentation and labeling has always been the most time-consuming task in medical image annotation. Conventionally, doctors of different professions and qualifications have to perform point-by-point manual segmentation and labeling for different imaging modes, anatomy structures, and lesions. The Pair team developed the intelligent function AutoSeg to implement refined segmentation and labeling of common anatomy structures. This significantly improves labeling efficiency and promotes the implementation and popularization of AI image research results.

The auto segmentation and labeling functions of the Pair software are implemented based on the MindSpore framework. The inference engine of MindSpore Lite is used to complete the inference deployment of the segmentation model, achieving a speed boost of 30% compared with the original LibTorch library. The high-performance operators provided by MindSpore Lite meet accuracy requirements using only common CPUs of common PCs. This effectively balances the compute power and accuracy requirements while also shortening the inference time to less than 1 second, greatly improving user experience of AutoSeg. The AutoSeg function of Pair shortens the segmentation and labeling time by about 70%.

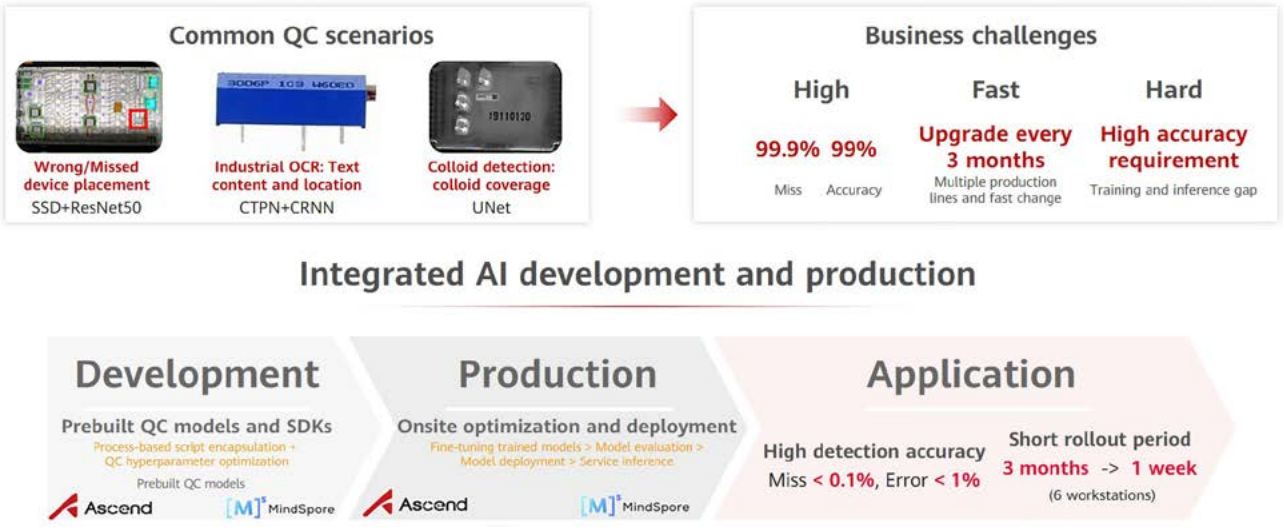


Figure 13 MindSpore-based intelligent manufacturing solution

MindSponge: Next-Generation AI Molecular Dynamics Simulation Software Framework

As an important theoretical research method, molecular simulation builds a bridge between the micromolecular world and the macroscopic observable measurement, thus providing a tool for us to understand the structure and dynamic properties of matter at the molecular level. Molecular simulation is widely used in sectors such as the chemical industry, biomedicine, energy, and materials. Despite the growth of this field aboard, there is few mature molecular simulation software with independent IP rights at home. In addition, there are still many technical, reliability, and efficiency problems in the applications of available molecular simulation software.

To address these problems, Peking University and Shenzhen Bay Lab Gao Yiqin Subject Team and Huawei MindSpore jointly developed MindSponge, a next-generation AI molecular dynamics simulation framework. This framework is a proprietary molecular simulation software library. It features high performance and modularization and allows scientists to efficiently and easily build computing modules required for molecular dynamics simulation. This framework can efficiently complete the molecular simulation process based on MindSpore features such as auto parallelism and graph kernel fusion (see Figure 15). With MindSpore's auto differentiation feature, MindSponge empowers traditional molecular simulation with AI methods and the high-performance computing feature of the MindSpore framework.

MindSponge aims to build the next-generation molecular simulation software platform that leads technology transformation in the computing era. It is China's first open-source general-purpose molecular simulation software framework that enables AI-powered molecular simulation.

4.3 Community Operations

Open source communities are booming in recent years. According to Gitee (China's dominant platform for hosting open-source code), the number of open-source projects increased by 192% in 2020 and China ranked top by the

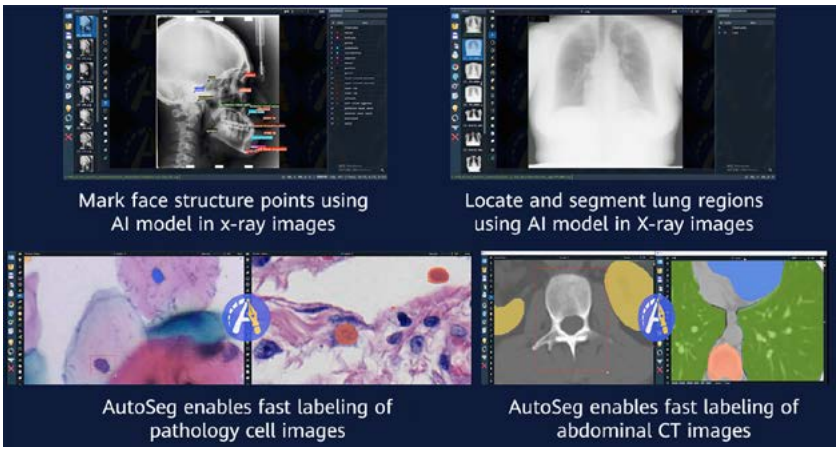
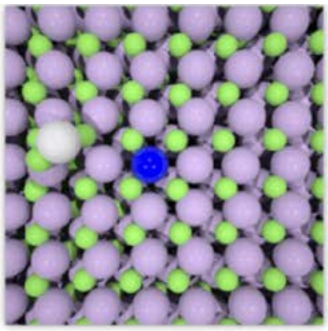


Figure 14 Pair for intelligent image labeling


QM/MM simulation with Gaussian and other quantization software


Simulation of ionic crystal surface

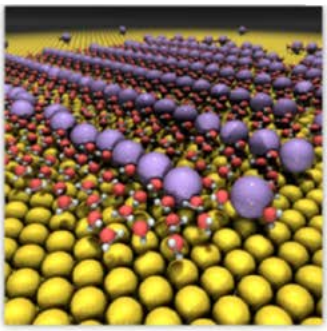

MindSponge's unique FGM simulation method that can better simulate the interface system

Figure 15 Typical applications of MindSponge

growth rate of open-source users. However, most open-source users have limited understanding of open-source and community culture, bringing severe challenges to open-source community governance. In this context, the MindSpore community, which focuses on AI tech trends, has been exploring effective community governance solutions to bring ultimate experience to community users and developers.

Community development activities centered on issues and pull requests (PRs) are a major part of community governance. The key to community development is fast issue closure and efficient code integration. The MindSpore community operation team surveyed the engineering practices of top communities in the industry and set up a developer experience special interest group (SIG) in the MindSpore community with Professor Cao Jian from the Shanghai Jiao Tong University. The SIG aims to empower community governance and operation with AI-based community robots (see Figure 16), reduce the threshold for developers, improve the governance efficiency of operations personnel, and enliven the community atmosphere.

Figure 17 shows an example of issue transfer. The community mentor tags historical data in order to generate learning samples, builds a model based on CNN/GAN text features, and uses the community robot to automatically recommend tags or developers, accelerating issue transfer. This service reduces the initial response time of community issues from 140 hours to 20 hours. Since the rollout in the MindSpore community, the robot has been called more than 80 times per day. In the next phase, MindSpore will focus more efforts in developer profiling and relationship graph building through developer feature modeling. In this way, MindSpore will identify active and inactive developers and improve the developer retention rate in the community.

5 Conclusions and Future Work

AI frameworks constitute the core of the deep learning infrastructure. The first-generation AI frameworks, represented by Torch, Theano, and Caffe, lay a foundation for Python-based deep learning network building, auto differentiation, and graph-based model representation and execution. Among the second-generation AI frameworks, TensorFlow has found wide industrial applications with its distributed training and diverse deployment capabilities while PyTorch has attracted the attention of scientists and algorithm engineers thanks to the flexibility brought by its dynamic graph mechanism. Standing on the shoulders of

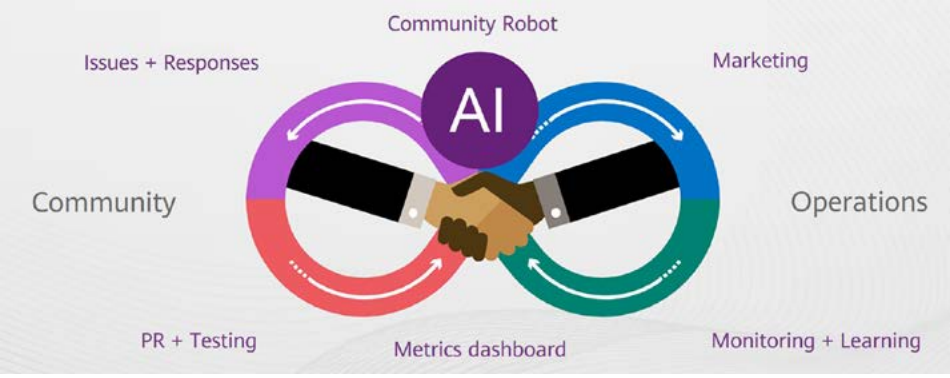


Figure 16 Robot-assisted governance of the MindSpore community

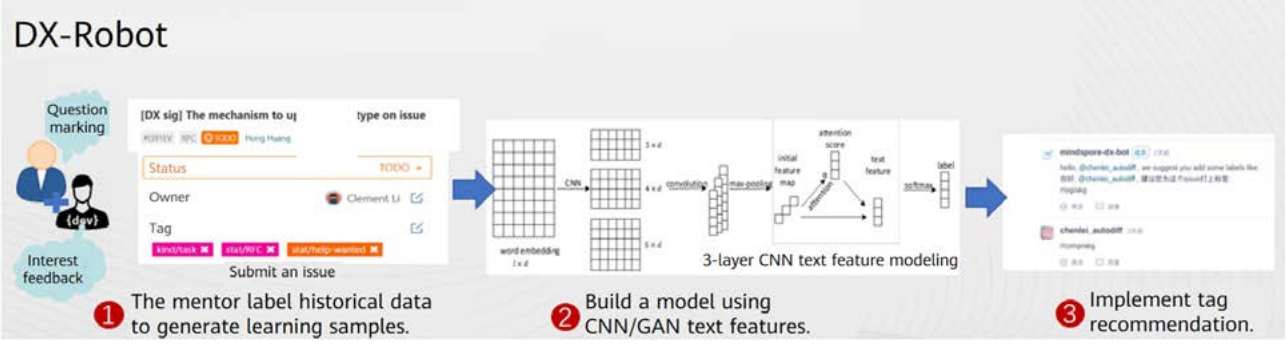


Figure 17 Issue transfer in the MindSpore community

giants, MindSpore builds its competitive advantages through a series of technology innovations.

Looking beyond the trends, driving factors, and challenges of AI development, we expect MindSpore to lead the evolution of AI frameworks in the following aspects:

1. Ultimate AI experience

To evolve from a deep learning framework to a generic tensor differentiable computation framework, MindSpore will become compatible with more big data analysis and scientific computing tools such as Pandas, NumPy, and SciPy, enabling scientific computing expressions to be simplified. To extend the application catalogs, MindSpore will offer a generic AI compiler that achieves a better trade-off between flexible model representation and efficient model execution. MindSpore will realize visualized intelligent optimization to simplify the inference and training process. And to maintain its advantages in cross-scenario collaboration, MindSpore will leverage innovative techniques such as performance/accuracy adaptation, heterogeneous parallelism, models, and code, as well as unified support for registration of proprietary and third-party chip drivers.

2. Ultra-large AI scale

With the development of brain-like science and feature learning, training of ultra-large-scale neural networks will revolutionize service performance and lead to the ever-increasing scale and complexity of data and models. To address these challenges, MindSpore will enhance hybrid parallelism on the basis of data parallelism and model parallelism, support elastic training and high-order optimization, and develop software/hardware co-optimization.

3. AI-based scientific computing

The industry has been exploring the combination of AI and scientific computing in the following three directions: First, replace traditional computing models with AI models, for example, the Deep Potential Molecular Dynamics (DeePMD) method. Second, employ AI in solving scientific computing equations, for example, PINNs and PINN-Net. Significant challenges still exist in this direction, especially in terms of accuracy and convergence. Third, accelerate equation solving with the framework. That is, the framework is considered a distributed framework oriented to tensor computing, and the same framework used for deep learning is applied in equation solving with the model and scientific computing method unchanged. To implement mathematical solvers for typical scientific computing problems, MindSpore will leverage high-order hybrid differentiation, multi-mode data convergence, and multi-scale hybrid computing methods. In this way, it will evolve the framework workload from deep learning models only to generic tensor differentiable computing.

4. Secure and trustworthy AI

As the bearer of AI services, AI frameworks must consider AI responsibilities, including security, privacy, fairness, transparency, and explainability. Current AI frameworks face the following challenges: lack of common analysis methods, measurement systems, and scenario-aware automatic measurement methods for every aspect of AI responsibilities; AI model robustness, privacy protection methods, and encrypted AI having a major impact on model performance; and lack of theoretical and algorithm support for AI explainability and human-friendly explanations of inference results. MindSpore will continue to build native secure and trustworthy AI capabilities including adversarial training, differential privacy, encrypted AI, federated learning, and explainable AI, in order to gradually evolve from consumer-level AI to enterprise-level AI.

Technology is at the root of a strong ecosystem. The MindSpore open source ecosystem will grow and thrive as AI technologies continue to evolve. The MindSpore technology ecosystem is committed to helping the AI industrial ecosystem, accelerating industrial implementations of AI technologies, and providing solid underlying support for the development of intelligent economy and digital transformation of more industries.

References

[1] A. Krizhevsky, L. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," In Advances in neural information processing systems, pp. 1097-1105, 2012.

[2] G. E. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal processing magazine, pp. 82-97, Nov. 2012.

[3] M. Volodymyr, K. Koray, S. David, A. R.Andrei and V. Joel, "Human-level control through deep reinforcement learning," Nature, vol. 518, pp. 529-533, 2015.

[4] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "A neural probabilistic language model," Journal of machine learning research, vol. 3, pp. 1137-1155, Feb. 2015.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: large-scale machine learning on heterogeneous distributed systems," 2016, arXiv:1603.04467. [Online].

[6] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in pytorch," In Advances in neural information processing systems, pp. 1-4, 2017.

[7] Y. Q. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: convolutional architecture for fast feature embedding," In Proceedings of the 22nd ACM international conference on Multimedia, pp. 675–678, ACM, 2014.

[8] T. Q. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Y. Zhang and Z. Zhang, "MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems," 2015, arXiv:1512.01274. [Online].

[9] G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn, *et al.*, "Dynet: the dynamic neural network toolkit," 2017. arXiv:1701.03980. [Online].

[10] S. Tokui, K. Oono, S. Hido and J. Clayton, "Chainer: a next-generation open source framework for deep learning," In Advances in neural information processing systems, pp. 1-6, 2015.

[11] N. Qian, "On the momentum term in gradient descent learning algorithms. Neural networks," the official journal of the International Neural Network Society, vol. 12, pp. 145–151, 1999.

[12] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," Journal of Machine Learning Research, vol. 12, pp. 2121–2159, 2011.

[13] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," International Conference on Learning Representations, pp. 1–13, 2015.

[14] M. Y. Chen, K. X. Gao, X. L. Liu, Z. D. Wang, *et al.*, "THOR, Trace-based hardware-driven layer-oriented natural gradient descent computation," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 7046-7054, 2021.

[15] T. B.Brown, B. Mann, *et al.*, "Language models are few-shot learners," 2020. arXiv:2005.14165. [Online].

[16] C. Leary and T. Wang. XLA: TensorFlow, compiled, 2017.

[17] T. Q. Chen and T. Moreau, "TVM: an automated end-to-end optimizing compiler for deep learning," the Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation, pp. 579-594, Oct, 2018.

[18] M. Li, Y. Liu, *et al.*, "The deep learning compiler: a comprehensive survey," 2020. arXiv:2002.03794.

[19] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and harnessing adversarial examples,"2014. arXiv:1412.6572. [Online].

[20] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," IEEE Access, vol. 6, pp. 14410–14430, 2018.

[21] A. Kurakin, I. Goodfellow and S. Bengio, "Adversarial examples in the physical world," 2016. arXiv:1607.02533. [Online].

[22] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," In 2017 IEEE symposium on security and privacy, pp. 39–57, 2017.

[23] M. Fredrikson, S. Jha and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1322–1333, 2015.

[24] R. Shokri, M. Stronati, C. Song and V. Shmatikov, "Membership inference attacks against machine learning models," In 2017 IEEE Symposium on Security and Privacy, pp. 3–18, 2017.

[25] T. Elsken, J. H. Metzen and F. Hutter, "Neural architecture search: a survey," 2018. arXiv:1808.05377. [Online].

[26] M. Wang, D. Zheng, *et al.*, "Deep Graph Library: a graph-centric, highly-performant package for graph neural networks," 2020. arXiv:1909.01315. [Online].

[27] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," CoRR, abs/1903.02428, 2019.

[28] J. Z. Huang, H. F. Wang, Y. B. Sun, M. Fan, Z. J. Huang, C. Y. Yuan, Y. W. Li, "HGAMN: heterogeneous graph attention matching network for multilingual POI retrieval at Baidu Maps," Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 3032-3040, 2021.

[29] R. Zhu, K. Zhao, *et al.*, "AliGraph: a comprehensive graph neural network platform," 2019. arXiv:1902.08730. [Online].

Huawei BiSheng Compiler Innovations and Practices

Yaoqing Gao [1], Baojian Hua [2]
[1] Compilers & Programming Languages Lab
[2] School of Software Engineering of USTC

Abstract

The ever-increasing service requirements of the computing industry, and the continuous improvement and innovation in the hardware architectures and micro-architectures, such as software-hardware co-design, software-defined hardware, and instruction set architecture (ISA) extension, keep posing new challenges to the research on compiler design and implementation. To cope with these challenges while addressing service pain points, Huawei BiSheng compiler, through a deep dive into the service requirements, research hotspots, and development trends in the industry and academia, introduces technological innovations at each stage of the legacy three-stage compiler architecture to build key differentiated competitiveness. With fast iterative evolution, the BiSheng compiler has been proven to empower Kunpeng and Ascend chips in delivering ultimate performance in typical scenarios, and is helping build a prospective technical ecosystem. In light of the future technology prospects, the BiSheng compiler is actively expanding and planning new innovative research directions. Based on insights into compiler technology evolution, this paper systematically introduces the technical innovations and practices of the BiSheng compiler, explains the key technical features, and predicts the technical directions for further evolution and innovation.

1 Preface

Compilers are important basic software. They translate source code written in a high-level programming language into a set of machine-language instructions. In the process of translation, a compiler performs complex analysis and optimization on the programs to improve their execution performance. Meanwhile, the compiler needs to balance the correctness of high-level programming language implementation and the complexity of underlying target architecture processing. High-level software, including the operating system (OS), database, network protocol stack, and artificial intelligence (AI) and machine learning (ML) frameworks, demands effective support from the underlying compiler. Owing to its important role in the entire software stack, compiler is valued as the crown jewel in software development.

Compiler is also a discipline that closely combines theory and practice in computer science. Centering on the theoretical research and engineering practice of compilers, the academia and industry have studied many compiler design theories and implemented a lot of effective engineering practices. Typical ones include programming language type, syntax [1–3], type system, lattice- and floating point-based data flow analysis [4–6], graph theory and application [7, 8], and constraint solving. Yet, some outstanding issues remain open in compiler design, such as back-end register allocation [9, 10] and instruction scheduling [11, 12], which are all deemed as non-deterministic polynomial-time hardness (NP-hardness) issues. Exploration of these issues is still a hot topic in compiler research.

There have also been remarkable achievements and progress in compiler engineering. In addition to the open source compilers widely used in open source communities, many universities, research institutes, and companies have maintained or released their own open source or commercial compilers for different languages and target architectures. These compiler engineering practices have played an important role in promoting academic research and supporting business success.

Along with the changing requirements in new application scenarios, evolution of hardware architectures, and development of basic theories and technologies, great progress has been made in theoretical research and innovation as well as practical exploration of compiler technologies. These new innovations in compiler design and implementation pave the way for new application scenarios such as big data, cloud computing, heterogeneous computing, and deep learning. However, there is no latest literature providing systematic introduction and summary on the new achievements and progresses of compiler design, as well as a systematic perspective on new directions.

To help address the lacuna, this paper is intended to systematically summarize and introduce compiler technology innovations and practices based on Huawei BiSheng compiler [13], and explore the future development directions. It covers the following topics: 1) A retrospective look into compiler development, architecture, and industry status, and a summary of important compiler-related concepts based on an in-depth analysis of the latest progress of the most popular open source LLVM compiler. 2) A deep dive into the architecture of Huawei BiSheng compiler, especially its advanced compilation optimization technologies, such as loop optimization and automatic vectorization, and a discussion on AI-assisted automatic compiler optimization. These features have made great contribution to the success of the BiSheng compiler. 3) An insight into the possible development directions and value-adds of Huawei BiSheng compiler in the future.

2 Introduction to Compilers

2.1 Basic Concepts

A compiler is a computer program that converts the source code written in one programming language (source language) into a program written in another equivalent programming language (target language). Its main purpose is to translate source programs written in high-level programming languages that are easy to write, read, and maintain into programs (that is, executable files) written in low-level machine languages that can be interpreted and run by computers. The source language is usually a high-level language, for example, C, C++, Java, Rust, or Go, whereas the target language is generally an assembly language or the target machine object code (also called machine code), for example, x86, ARM, or RISC-V [14].

The compiler structure can be logically divided into two major parts: front-end and back-end. There is a clear division of responsibilities between them. The front-end focuses on understanding of the source program. It divides the source program into multiple lexical units, checks

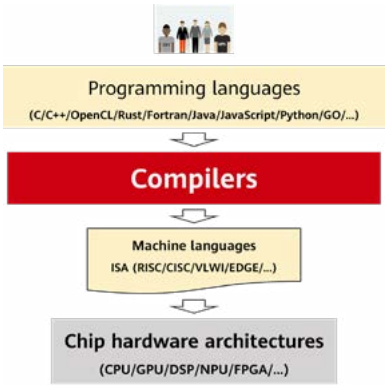


Figure 1 Compiler use case

whether the lexical units meet the syntax structure specified by the language and whether the source program complies with the program semantics, and generates an intermediate representation (IR).

The back-end focuses on mapping of the IR generated by the front-end to the target machine. Its major tasks include selecting the appropriate machine instructions for the syntax structure of the source program, allocating the appropriate resources of the target machine for the variables of the source language, and so on.

In modern optimizing compilers, an additional part, namely, the mid-end, is introduced. It is an intermediate phase that is relatively independent of a specific programming language and target machine. The introduction of the mid-end brings two benefits: First, the front-end and the back-end are decoupled. This allows the front-end of multiple languages to generate common mid-end code, which can be further used to generate different target machine object code. Second, program optimization algorithms that are irrelevant to programming languages and target machines can be designed and implemented at the mid-end to optimize program performance, scale, or other metrics.

Regarding the high complexity and large-scale application of compilers, the compiler design should be able to address the productivity, performance, and portability (3P) challenge from the following perspectives:

- Improve programming **productivity** of application developers by shielding hardware architectures to facilitate efficient programming.
- Enable ultimate hardware **performance** through static and dynamic compilation optimization and software-hardware synergy.
- Improve software **portability** by porting software from open source communities to solve the hardware-specific cross-generation source code and binary compatibility issues as well as the compiler upgrade-specific software behavior and performance compatibility issues.

In addition, the compiler should be able to ensure the security and reliability of program code, and to implement error detection and correction by means of dynamic and static program analysis.

Due to the importance of compiler theories and practices, the compiler has always been at the core of computer science. Take the Turing Award, the highest award in computer science, as an example. The first award was presented to professor Alan Perlis, a programming language and compiler expert. Since then, there have been awards for programming language compiler research every three to five years on average.

Year	Author	Awarded for	Remarks
2020	Jeffrey Ullman, Alfred Aho	Basic compiler theory and algorithm	
2017	Hennessy, John L, Patterson, David	Hardware structure, compiler	
2013	Lamport, Leslie	Concurrent programming	
2008	Liskov, Barbara	Programming language, system design	
2006	Allen, Frances ("Fran") Elizabeth *	Compilation optimization, automatic parallelism	IBM compiler development team
2005	Naur, Peter *	Programming language design	
2003	Kay, Alan	Object-oriented language Smalltalk	
2001	Dahl, Ole-Johan *Nygaard, Kristen *	Object-oriented language Simula	
1999	Brooks, Frederick ("Fred")	Hardware architecture, software engineering	
1987	Cocke, John *	RISC architecture, compilation optimization	
1986	Hopcroft, John ETarjan, Robert (Bob) Endre	Formal language theory	
1984	Wirth, Niklaus E	Programming language, compilation system, architecture	
1980	Hoare, C. Antony ("Tony") R.	Programming language, concurrency theory	
1979	Iverson, Kenneth E. ("Ken") *	Programming language APL	
1977	Backus, John *	Programming language Fortran, compiler	IBM compiler development team
1972	Dijkstra, Edsger Wybe *	Program design	
1966	Perlis, Alan J *	Program design, compilation structure	

Figure 2 Turing Award for compiler-related research

2.2 Development History

Since the mid-1950s, compiler design has been an important research field in computer science. The Fortran compiler developed by IBM is the first widely used high-level language compiler. It is a multipass system and has introduced many important concepts of modern compilers in its design and implementation. It is a multipass system and has introduced many important concepts of modern compilers in its design and implementation, including independent lexical analyzer, syntax analyzer, and program optimization algorithms like register allocation.

In the 1960s and 1970s, researchers studied and built many influential compilers. Among them are the classic optimizing compiler FORTRANH [15, 16], Bliss-11 and Bliss-32 compilers [17], and portable BCPL compilers [18], to name just a few. These compilers can generate high-quality object code for various complex instruction set computer (CISC) architectures.

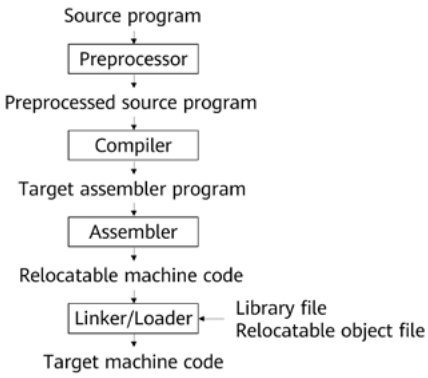


Figure 3 Typical compiler architecture

In the 1980s, the advent of reduced instruction set computer (RISC) architecture had a profound impact on compiler design. This architecture requires compiler designers to track the characteristics of new programming languages and study and design new compilation algorithms to unleash the computing power of new hardware. These trends forced the new generation of compilers [19–21] to focus on more powerful mid-end code optimization, as well as back-end code optimization and generation technologies. The typical architecture of compilers in this phase is as illustrated in the following figure. Modern RISC-based compilers still follow this architecture model.

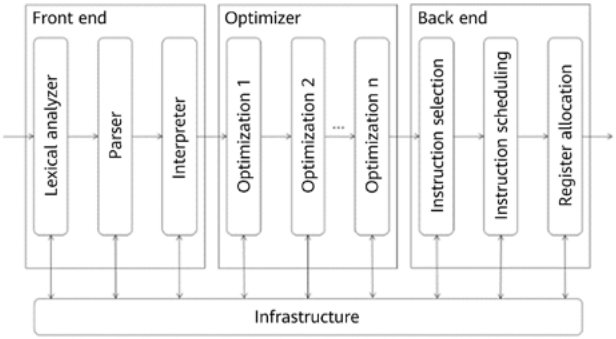


Figure 4 Typical architecture of optimizing compilers

Since the 1990s, compiler researchers turned their focus to address the challenges posed by the microprocessor architecture, including multifunctional processor components, memory latency, and code parallelization. In fact, the compiler structure and organization proposed for the RISC architecture in the 1980s are still flexible enough to cope with these challenges. Researchers can construct new passes and insert them into the compiler's optimizer and code generator to resolve the challenges posed by the architecture [22].

As the cost of independently developing proprietary closure compilers became increasingly high, Richard Matthew

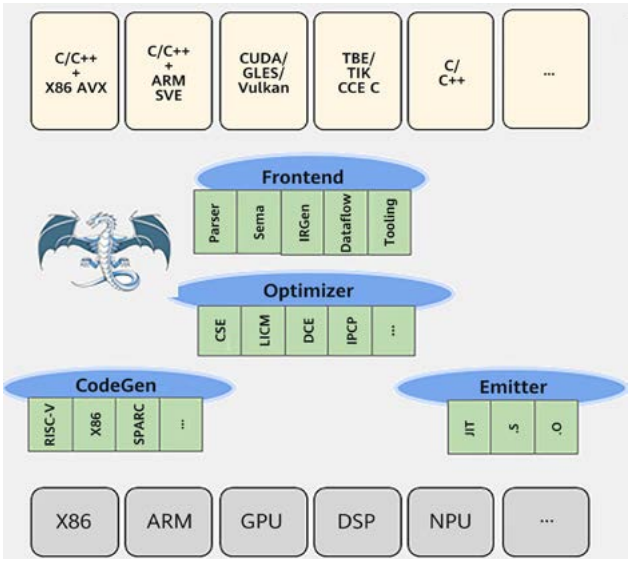


Figure 5 LLVM architecture

Stallman decided to develop an open source GCC compiler, which turned out to have a far-reaching impact on the future development of compilers. It is the official compiler of GNU systems (including GNU and Linux families) and also a key compiler for other OSs, including the BSD family, Mac OS X, NeXTSTEP, and BeOS.

GCC with a three-stage architecture has proven to be the first choice for cross-platform software compilers. Unlike compilers that are limited to specific systems and operating environments, GCC uses the same front-end processing program on all platforms. This way, it produces the same IR that can be compiled on various other platforms and outputs the compiled program correctly [23].

2.3 LLVM Compiler

As the system architecture and programming language become more diverse, GCC is not flexible enough to quickly adapt to new requirements and build competitiveness. The open source LLVM compiler is based on the three-stage architecture and uses a modular design and library implementation. Its ecosystem compatibility, friendly software licensing, community activeness, and learning cost are popular in the academia and industry, making LLVM a dominant trend in the compiler field.

LLVM applies to multiple programming languages and target processors. It is written in C++ and consists of the front-end, back-end, optimizer, as well as some library functions and modules. It is open source and compatible with existing scripts written for GCC. LLVM has been

Company	Compiler	Application Scenario
Apple	Swift/Objective-C	Consumer device
Google	Android compiler toolchain	Consumer device
Qualcomm	Android compiler + Hexagon compiler	Consumer device+5G
IBM	XL C/C++ compiler	High-performance computing/General-purpose server
AMD	AOCC Fortran/C/C++ compiler	High-performance computing/General-purpose server
ARM	AHC compiler/Mali GPU compiler	Consumer device/High-performance computing/General-purpose server
Intel	DPC++ compiler	High-performance computing/General-purpose server
Others	Facebook, Microsoft, Nvidia, Xilinx, Synopsys....	

Figure 6 A glimpse of LLVM applications

recognized as a common architecture for implementing various static and runtime compilation languages. Currently, more than 140 companies have participated in the open source LLVM project, spanning a wide range of industries and hardware architectures. The LLVM community has maintained a high popularity in the past decade.

Major contributors include Apple, Google, and key chip vendors. With LLVM, they are able to support a variety of standard languages and self-developed languages, such as Swift, Rust, and Go, and empower chips with different architectures, such as x86, ARM, and RISC-V. In addition, they have developed some innovative technologies, for example, multi-level intermediate representation (MLIR) [21, 24].

Currently, key industry players have joined the LLVM project. In addition to contributing to the community, they build their own closed source compilers based on the LLVM architecture by using highly competitive compilation technologies. Apple, Google, Qualcomm, IBM, AMD, ARM, Intel, Facebook, Microsoft, and Huawei are backers for LLVM in diverse scenarios, including consumer device, 5G, general computing, high-performance computing (HPC), and server.

3 Huawei BiSheng Compiler

3.1 Introduction

BiSheng compiler is a Huawei-developed compiler that delivers high performance and diversified computing power. Based on the open source LLVM compilation framework [25], it provides common compilation optimization technologies with differentiated competitiveness to cater to diverse application scenarios, programming languages, and hardware architectures, and implements in-depth collaborative optimization specific to different architectures and application scenarios. Figure 7 is the overall architecture of the BiSheng compiler. The compiler converts programs

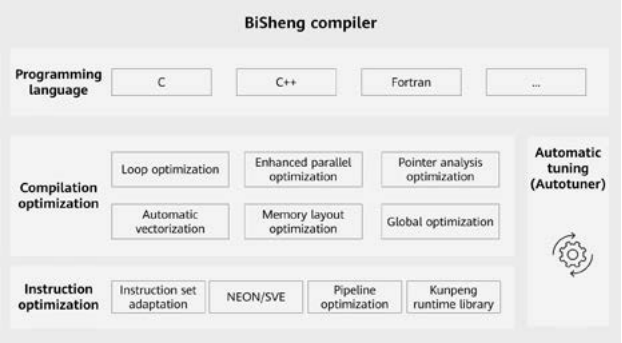


Figure 7 Architecture of the BiSheng compiler

written in different programming languages into unified internal IRs. It then carries out a series of pass optimizations, such as loop optimization, memory layout optimization, and automatic vectorization, to implement IR optimization and conversion, explore the parallelism and locality in the programs, and make full use of hardware features like built-in acceleration instruction. With the built-in Autotuner, the compiler automatically tunes the optimization parameters that cannot be determined during static optimization. Finally, it outputs high-performance executable programs targeted for the Kunpeng platform.

3.2 Key Competitive Technologies

The BiSheng compiler focuses on key competitiveness in multiple fields, such as high-performance compilation algorithm, custom acceleration instruction set optimization, and AI-assisted automatic tuning (Autotuner). Specifically, the compiler explores parallelism and locality in programs by using advanced high-performance compilation algorithms, selects proper custom instruction set acceleration programs through architecture-specific instruction optimization, and automatically tunes optimization parameters, which cannot be determined during static optimization, through AI-assisted automatic tuning.

3.2.1 High-Performance Compilation Optimization Algorithms

Traditional chips seek for universality and therefore are designed with complex microstructures. This hinders the software from bringing hardware performance into full play. To mine the potential of transistors in specific scenarios, on the premise that performance and power consumption are prioritized, targeted optimization can be performed based

on domain characteristics to improve the performance and power consumption ratio at the cost of software complexity. This is the main approach to further unleash performance in the industry.

David Patterson and John Hennessy, two heavyweights in the computer architecture field, in their speech at the Stanford University in 2017, highlighted that the program performance can be greatly improved by blending multiple architecture designs and compilation optimization algorithms [26]. Figure 8 shows the performance acceleration ratios of matrix multiplication computing under different implementations and optimization technologies.

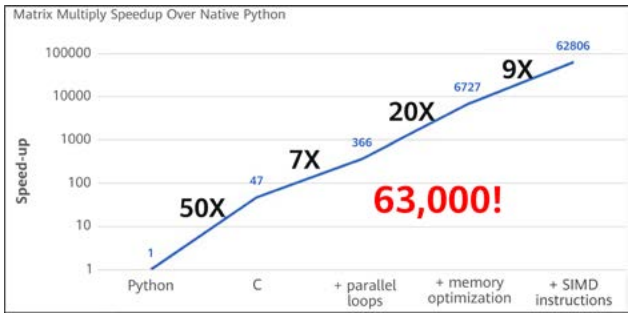


Figure 8 Performance acceleration ratios of matrix multiplication computing

To achieve better performance, the BiSheng compiler also focuses on the following aspects to make full use of hardware:

1. More effective parallelism for specific domains at different hierarchies
   - Loop-level parallelism
   - Instruction-level parallelism (ILP) and memory-level parallelism (MLP)
   - Superscalar, multi-thread, multi-core, single instruction, multiple data (SIMD), and single thread, multiple data (STMD)
   - Very long instruction word (VLIW) versus speculation and disorder
2. More efficient memory usage in the case of space/time locality
   - User control vs high-speed cache
   - Cache access, memory transfer, and function block division
3. Elimination of unnecessary data precision
   - Substitute of IEEE floating point arithmetic with low-precision floating point arithmetic
   - Conversion from an integer of 32–64 digits to an integer of 8–16 digits
   - Hybrid-precision calculation mode

From the preceding aspects, the BiSheng compiler has made targeted improvements to the compilation optimization algorithms.

3.2.1.1 Parallel Loop Optimization

Statements in a loop are usually executed for multiple times and hence loop-dedicated optimization is expected to yield significant benefits. Loop optimization is an important means to improve the compiler performance, and it involves a wide range of optimization algorithms. As mentioned in *A new golden age for computer architecture* [26], loop parallelization can greatly improve performance in applicable scenarios. By utilizing different algorithms (such as vectorization and instruction scheduling algorithms), a compiler achieves significant improvement in loop optimization effects, including improved cache usage, reduced register pressure, fewer dynamic instructions, and reuse of load actions or computing values in different iterations to expose other optimization opportunities.

The BiSheng compiler is built with multiple loop optimization methods. Important optimization technologies used by the BiSheng compiler include:

1. Loop Unrolling and Jamming

This technology expands the outer loop of the nested loop and integrates the loop body of the inner loop to improve the cache usage, as shown below:

```
for (int i = 0; i < n; i++) {
  for (int j = 0; j < m; j++) {
    a[j] += b[i][j] + b[i-1][j];
  }
}
```

After compiler optimization:

```
for (int i = 0; i < n-n%2; i+= 2) {
  for (int j = 0; i < m; j++) {
    a[j] += b[i][j] + b[i-1][j];
    a[j] += b[i+1][j] + b[i][j];
  }
}
```

Through the preceding conversion, the cache hit ratio is effectively improved.

2. Loop Fusion/Distribution

Loop Fusion combines the loop bodies of two loops in an attempt to create a new loop. The advantage of this approach is to identify reusable values and expose instruction scheduling opportunities.

Loop Distribution is the reverse process of Loop Fusion. It divides a loop body into two independent loop bodies. The advantage of this approach is to expose vectorization optimization opportunities and seek additional optimization space.

In the following example, the first two statements are saved together in order to reuse the loaded values while separating a third statement that cannot be vectorized.

```
for (int i = m; i < n; i++) {
  a[i] = b[i-1] + b[i];
  c[i] = b[i-1] - b[i];
  d[i-1] = d[i-2] + 1;
}
```

After compiler optimization:

```
for (int i = m; i < n; i++) {
  a[i] = b[i-1] + b[i];
  c[i] = b[i-1] - b[i];
}

for (int i = m; i < n; i++) {
  d[i-1] = d[i-2] + 1;
}
```

3. Loop Unrolling

This technology decreases loop iterations by copying loop bodies and hence reduces branch jumps and loop boundary checks. In addition, the expanded loop body can generate new reusable values, helping expose more instruction scheduling opportunities.

```
for (int i = 0; i < n; i++) {
  a[i] = b[i-1] + b[i];
}
```

After compiler optimization:

```
int ub = n - n%2;
for (int i = 0; i < ub; i += 2) {
  a[i] = b[i-1] + b[i];
  a[i+1] = b[i] + b[i+1];
}
if (i < n)
  a[i] = b[i-1] + b[i];
```

4. Loop Unswitching

This technology extracts conditional branch statements that do not change conditions in a loop to reduce branch jumps and provide more local optimization opportunities.

```
for (int i = 0; i < n; i++) {
  a[i] = 1;
  if (b[j] > 10)
    a[i] += 1;
}
```

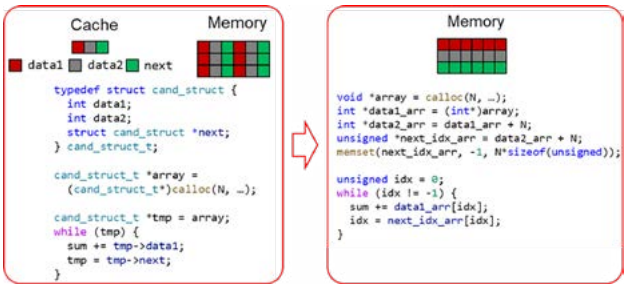
After compiler optimization:

```
if (b[j] > 10)
  for (int i = 0; i < n; i++)
    a[i] = 1;
else
  for (int i = 0; i < n; i++)
    a[i] = 2;
```

3.2.1.2 Memory Optimization

1. Structure memory layout optimization

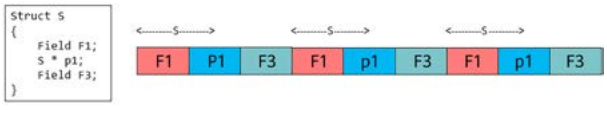
To improve the cache usage, the BiSheng compiler has enhanced structure memory layout optimization by way of whole-program optimization. The primary means is to convert structure arrays into array structures. Structures can be explicit, or can be inferred by checking array usage in a loop.



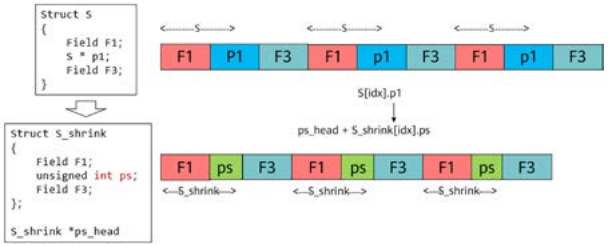
2. Structure pointer compression optimization

This technology improves the D-cache hit ratio by reducing the memory usage space.

The following example illustrates how the structure optimization technology works. As shown in the following figure, the member pointer P1 occupies eight bytes.



The domain member pointer is compressed and then extracted to reduce the memory size of each structure node. This increases the structure data that can be stored in a cache line and therefore improves the cache hit ratio.



The original domain member pointer is accessed by converting the global structure pointers ps_head and ps to the combination of the base address and relative offset.

3. Prefetch optimization

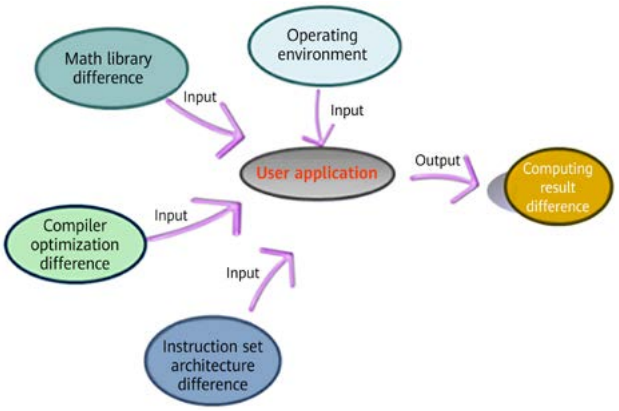
Software prefetch is an optimization technology that reads required data from the memory in advance by inserting prefetch instructions. The prefetch effect depends on the lead time. If data arrives too early, it will waste the cache space. If data arrives too late, it will need to wait for access. Therefore, factors including the cache line size, memory access delay, and loop size must be considered while determining the lead time of software prefetch. To address this requirement, the BiSheng compiler adjusts the software prefetch configuration specifically for the Kunpeng microarchitecture and selects accurate prefetch time to improve the D-cache hit ratio.

```
for (int i = 0; i < N; i++) {
    sum += a[i] * b[i];
}
```

```
for (int i = 0; i < N; i++) {
    prefetch(&a[i+k]);
    prefetch(&b[i+k]);
    sum += a[i] * b[i];
}
```

3.2.1.3 Floating-Point Precision Tuning

After HPC-based applications, such as Weather Research and Forecasting (WRF) and Global/Regional Assimilation and Prediction System (GRAPES) are migrated to Kunpeng servers, it is probable that the computing results are inconsistent with the original data. As shown in the following figure, the precision differences of computing results are mainly due to four factors: math library, compiler optimization, instruction architecture, and operating environment.



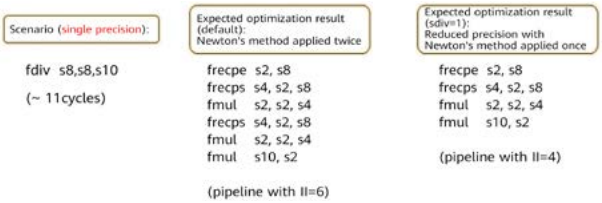
Sources of precision differences

The solutions to the precision differences are as follows:

- Compiler optimization: Support precision control options such as fp-model and optimize the compiler optimization behavior and precision.
- Instruction architecture: Analyze instruction differences using the transcoding tool and eliminate the differences using methods such as soft floating-point simulation.
- Math library: Rewrite functions in the math library, or use math libraries of different precisions.
- Operating environment: Support cluster and multi-kernel environments, such as MPI and OpenMP.
- Computing result: Optimize and improve the floating-point compilation algorithm in different precision modes.

Scenario 1:

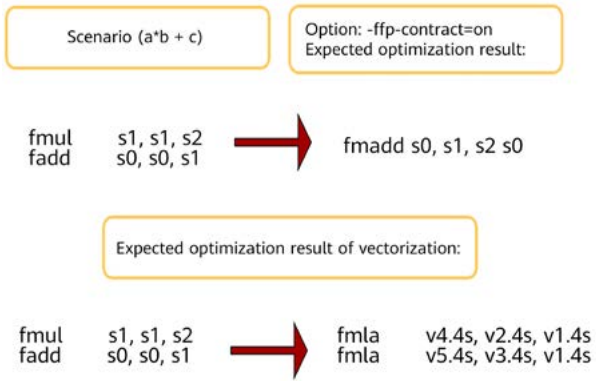
- Replacing division with multiplication to enable pipeline
- Adapting Newton's method to support on-demand precision adjustment



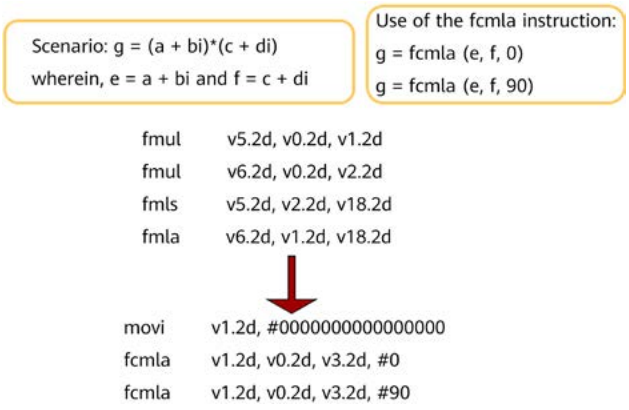
Scenario 2:

- Combining (complex number) multiply-add instructions to support dynamic precision optimization

Combination of multiply-add instructions



Combination of complex number multiply-add instructions

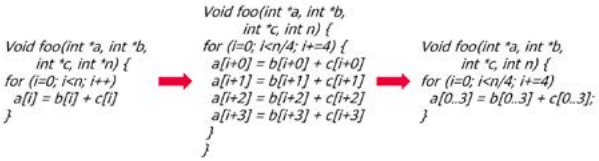


3.2.2 Acceleration Instruction Set Optimization

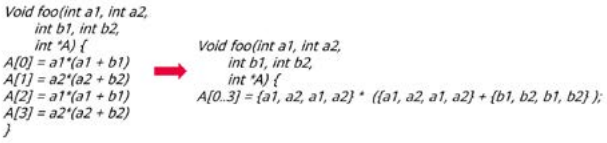
Customizing acceleration instructions is an important means of building chip competitiveness. The compiler plays an important role in optimization of instruction generation and instruction selection. One commonly used approach is vectorization optimization.

In parallel computing, automatic vectorization is a special case of automatic parallelization, where a program transforms from a scalar implementation that processes one pair of operands at a time, to a vector implementation that processes multiple pairs of operands simultaneously in one operation. For example, the AArch64-based Kunpeng 920 processor has thirty-two 128-bit vector registers and therefore allows the program to simultaneously operate four channels of 32-bit data or two channels of 64-bit data. The BiSheng compiler focuses on loop vectorization and superword-level parallelism (SLP) vectorization optimizations, which aim to ensure program locality while efficiently improving performance in computing-intensive scenarios.

The following figure depicts how loop vectorization works.

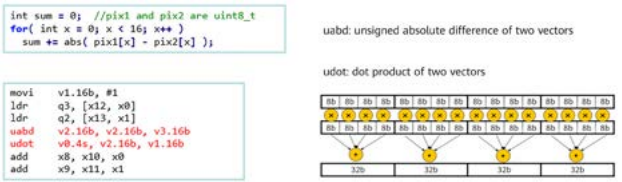


The following figure depicts how SLP vectorization works.

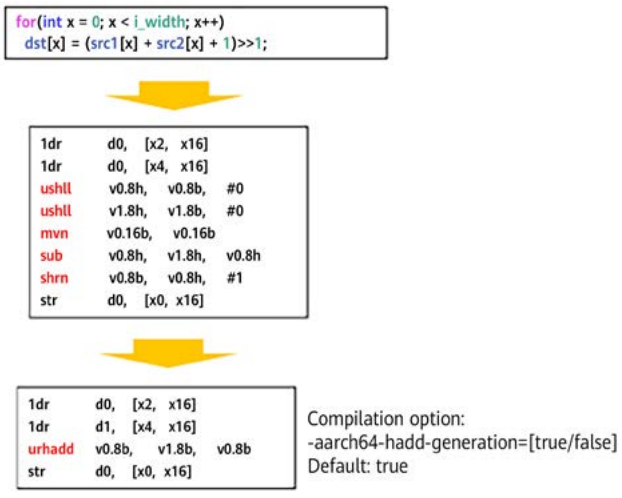


As shown in the figures, the BiSheng compiler automatically generates Kunpeng vector instructions through automatic vectorization. The following are two typical application scenarios of automatic vectorization optimization.

Scenario 1:



Scenario 2:



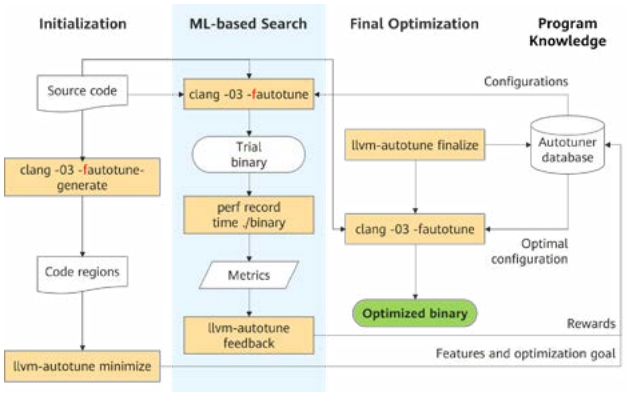
3.2.3 Automatic Tuning (Autotuner)

Automatic tuning is an automated iteration process that optimizes a given program by manipulating compilation options to achieve the optimal performance. According to

earlier researches, ML-based automatic tuning has been proven to outperform manual tuning.

The AI-assisted automatic tuning of the BiSheng compiler is implemented by the BiSheng compiler together with the Autotuner command line tool.

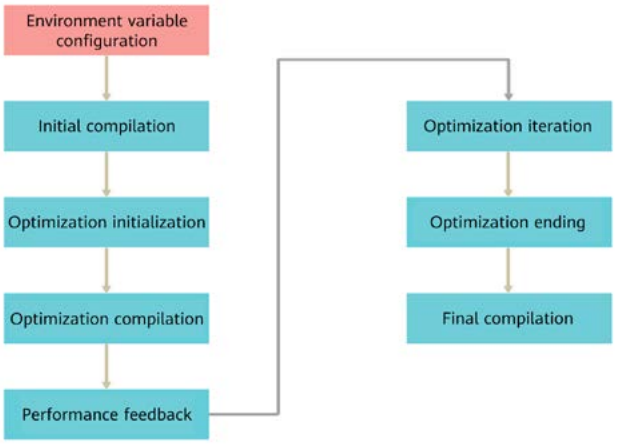
- The BiSheng compiler has a built-in automatic tuning feature. It can invoke Autotuner to implement optimization in a finer granularity.
- Autotuner is a command line tool. It generates the parameter search spaces, produces different combinations of parameters, and drives the entire tuning process.



Logical architecture of Autotuner

The ML-based search technology introduced by Autotuner boasts the following features:

1) Autotuner database: a knowledge base established based on static and dynamic analysis information to support decision-making system optimization.
2) Optimal configuration: a process of determining optimization measures based on hotspot and performance evaluation information and knowledge base information.
3) Performance evaluation: a process of hotspot marking, bottleneck detection, and performance measurement.
4) Search-driven feedback: a process of sending optimization suggestions provided by the optimization decision system to the compiler for implementing optimization.



Autotuner service process

The key steps of Autotuner service process are as follows:

1) Use the environment variable **AUTOTUNE_DATADIR** to specify the storage location of optimization-related data.
2) Add the compiler option **-fautotune-generate** to complete initial compilation and generate a tuning task.
3) Run the **llvm-autotune** command to initialize the tuning task. Keep the generated initial compilation configuration file for next compilation.
4) Add the compiler option **-fautotune** to read the current configuration in **AUTOTUNE_DATADIR** and perform compilation.
5) Run the program to obtain performance data based on your requirements. Then, run the **llvm-autotune feedback** command to request performance data feedback.
6) Based on the specified number of iterations, repeat the optimization compilation and performance feedback steps for optimization iteration.
7) After multiple rounds of iteration, end the optimization and save the optimal configuration file.
8) Use the optimal configuration file obtained in the previous step for final compilation.

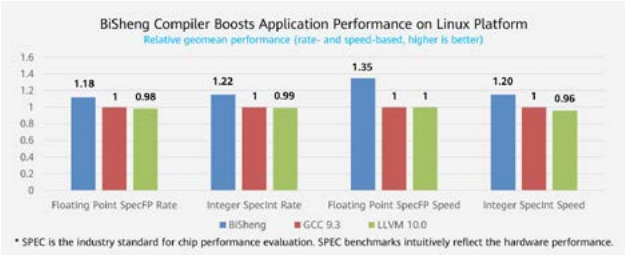
3.3 Optimization Effects

Through synergy with the Kunpeng chip, the BiSheng compiler maximizes the chip performance to deliver exceptional service experience on the Kunpeng platform.

This section presents the test results of the BiSheng compiler in different benchmarking and application scenarios.

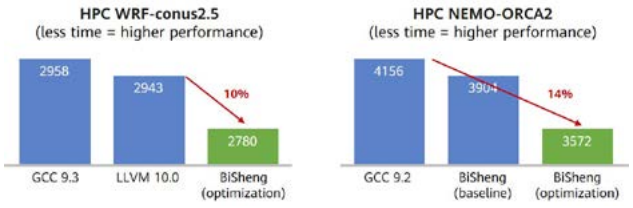
3.3.1 Industry CPU Benchmarking

The following figure shows the performance delivered by the same test program that has been optimized by the BiSheng compiler, GCC 9.3 compiler, and LLVM 10.0 compiler on the Kunpeng platform. According to the test results, the BiSheng compiler delivers performance outcomes better than the other two compilers, achieving an average performance increase by more than 20% compared with SPEC 2017.



3.3.2 Performance Improvement for Typical HPC Applications

As shown in the following figure, by working with the Kunpeng chip, the BiSheng compiler effectively improves performance experience in typical HPC applications.



4 Future Innovation and Evolution Directions of the BiSheng Compiler

The BiSheng compiler will seek all-round technological innovation from the following perspectives: basic theory innovation, software-hardware synergy architecture innovation, and service scenario-based precise tuning. Key research directions include:

1. Continuous program optimization (CPO): Compilation optimization is no longer limited to the development phase. The academia and industry are exploring high-performance runtime optimization technologies. In the future, CPO will remain an important direction for performance optimization. A typical example is the low-cost program runtime performance monitoring technology proposed by IBM CPO, which analyzes behavior changes of running programs to identify opportunities that cannot be accurately predicted and optimized by static compilation [27]. Another example is the libVC dynamic real-time compilation framework put forward by Milan Polytechnic, which streamlines CPO throughout the software lifecycle, from optimization of offline compilation in the development state to optimization of online compilation in the running state [28]. Although these technologies probe into different optimization strategies, they all employ CPO to implement real-time optimization throughout the program lifecycle.

2. AI for compiler: The application of AI in compilation optimization has been widely studied by large companies. In the future, AI-assisted compilation optimization will be an important direction for compilation optimization. For example, Intel and UC Berkeley are embarking on a joint research of using deep reinforcement learning (DRL) to implement decision-making optimization and automation for loop vectorization. According to the research, embedding a stable prediction model into the automatic vectorization phase of compilation helps obtain an optimized vectorization result [29]. Similarly, Google proposes the AI-assisted inlining optimization strategy, which uses the ML model to replace the heuristic algorithm of inliner to determine whether to perform inlining for more refined optimization.

3. System-on-chip (SoC) compiler: SoC chips have become the main development trend in the industry. It is quite probable that SoC compilers will become a new research direction. Some typical examples are Apple's M1 chip, PC processor made as mobile phone SoC chip, N-in-one chip, and Intel Alder Lake processor. They use a single, highly scalable SoC architecture involving new technologies such as unified memory architecture (UMA) and hybrid SoC architecture featuring Performance cores and Efficient cores, making it a necessity to explore matching compilation technologies.

4. Super optimizer: The academia has been trying to extend the research on basic compiler theories. In recent years, super optimizers have been accelerating their move from theory research to industry application.

They transform optimization problems into spatial search problems, and are exploring new algorithms to conduct recursive search within the finite search space of the program characteristic sequence (ISA/IR), so as to find the optimal program that can provide equivalent outputs. For example, technologies such as Stanford's STOKE [30] and MTK's Souper [31] have been applied to the LLVM compiler.

5 Conclusion

The BiSheng compiler builds key competitiveness by using different compilation optimization methods to cater to different chips, application scenarios, and application characteristics. It takes multiple factors, such as performance benefit, code volume, compilation time, and debuggability, into consideration during compilation optimization and unleashes ultimate computing power of chips through software-hardware synergy.

References

[1] Melvin E. Conway, "Design of a separable transition-diagram compiler," *Commun. ACM*, 6(7), pp. 396–408, 1963.

[2] Philip M. Lewis II and Richard Edwin Stearns, "Syntax-directed transduction," *J. ACM*, 15(3), pp. 465–488, 1968.

[3] Donald E. Knuth, "On the translation of languages from left to right," *Inf. Control*, 8(6), pp. 607–639, 1965.

[4] Allen, F. E., "Control flow analysis," *SIGPLAN Notice,* 5(7), pp. 1–19, 1970.

[5] Gary A. Kildall, "A unified approach to global program optimization," *POPL* 1973, pp. 194–206.

[6] Barry K. Rosen, "High-level data flow analysis," *Commun. ACM*, 20(10), pp. 712–724, 1977.

[7] G. J. Chaitin, "Register allocation and spilling via graph coloring," *United States Patent 4,571,678*, February 1986.

[8] G. J. Chaitin, "Register allocation and spilling via graph coloring," in *Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction*, *SIGPLAN Not.*, 17 (6), pp. 98–105, 1982.

[9] R. Lo, F. Chow, R. Kennedy, S. Liu, and P. Tu, "Register promotion by sparse partial redundancy elimination of loads and stores," in *Proceedings of the ACM SIGPLAN '98 Conference on Programming Language Design and Implementation*, *SIGPLAN Not.*, 33 (5), pp. 26–37, 1998.

[10] A. V. S. Sastry and R. D. C. Ju, "A new algorithm for scalar register promotion based on SSA form," in *Proceedings of the ACM SIGPLAN' 98 Conference on Programming Language Design and Implementation*, *SIGPLAN Not.*, 33 (5), pp. 15–25, 2008.

[11] Keith D. Cooper and Philip J. Schielke "Non-local instruction scheduling with limited code growth," *LCTES 1998*, pp. 193–207.

[12] Jack L. Lo and Susan J. Eggers, "Improving balanced scheduling with compiler optimizations that increase instruction-level parallelism," *PLDI 1995*, pp. 151–162.

[13] Edward S. Lowry and C. W. Medlock, "Object code optimization," *Commun. ACM 12(1)*, pp. 13–22, 1969.

[14] Randolph G. Scarborough and Harwood G. Kolsky, "Improved optimization of FORTRAN object programs," *IBM J. Res. Dev. 24(6)*, pp. 660–676, 1980.

[15] R. G. G. Cattell, Joseph M. Newcomer, and Bruce W. Leverett, "Code generation in a machine-independent compiler," *SIGPLAN Symposium on Compiler Construction 1979*, pp. 65–75.

[16] Martin Richards, "The portability of the BCPL compiler," *Softw. Pract. Exp. 1(2)*, pp. 135–146, 1971.

[17] Marc A. Auslander and Martin Hopkins, "An overview of the PL.8 compiler," *SIGPLAN Symposium on Compiler Construction 1982*, pp. 22–31.

[18] John Cocke and Peter W. Markstein, "Measurement of programming improvement algorithms," *IFIP Congress 1980*, pp. 221–228.

[19] Mark Scott Johnson and Terrence C. Miller, "Effectiveness of a machine-level, global optimizer," *SIGPLAN Symposium on Compiler Construction 1986*, pp. 99–108.

[20] John Hennessy and David Patterson, "A new golden age for computer architecture," Stanford and UC Berkeley, 13 June 2018.

[21] Chris Lattner, "The golden age of compilers," *ASPLOS 2021*, April 19, 2021.

[22] Ameer Haj-Ali, Nesreen K. Ahmed, Ted Willke, Sophia Shao, Krste Asanovic, and Ion Stoica, "Neuro vectorizer end-to-end vectorization with deep reinforcement learning."

[23] Abhijat Vichare, "GCC Internals: A Conceptual View," http://kcchao.wikidot.com/gcc-internals, Dec 2007.

[24] Chris Lattner, "MLIR primer: a compiler infrastructure for the end of Moore's law," *CGO*, 2019.

[25] David Chisnall, "What LLVM can do for you," *LLVM Developers' Meeting,* April 13, 2012.

[26] David Patterson and John Hennessy, "A new golden age for computer architecture," *IBM Journal of Research and Development*, pp. 239–248, 2006.

[27] C. Cascaval, E. Duesterwald, P. F. Sweeney, and R. W. Wisniewski, "Performance and environment monitoring for continuous program optimization," *ACM*, 2019.

[28] S. Cherubin and G. Agosta, "LIBVERSIONINGCOMPILER: An easy-to-use library for dynamic generation and invocation of multiple code versions," *SoftwareX*, pp. 95–100, 2018.

[29] Nesreen Ahmed, Ted Willke, Sophia Shao, Krste Asanovic, and Ion Stoica, "NeuroVectorizer: end-to-end vectorization with deep reinforcement learning," *International Symposium on Code Generation and Optimization 2020 (CGO 2020)*, February 2020.

[30] Eric Schkufza, Rahul Sharma, and Alex Aiken, "Stochastic Superoptimization," *SIGPLAN*, 2013.

[31] Raimondas Sasnauskas, Yang Chen, Peter Collingbourne, Jeroen Ketema, Gratian Lup, Jubi Taneja, and John Regehr, "Souper: a synthesizing superoptimizer," *Computer Science*, 2017.

Recent Advances in Online Matching

Zhihao Tang [1], Yuhao Zhang [2]
[1] ITCS, Shanghai University of Finance and Economics
[2] Shanghai Jiao Tong University

Abstract

Since the seminal work of Karp, Vazirani, and Vazirani [36], online matching stands as a cornerstone in the online algorithms literature. In 2013, Mehta [45] wrote a comprehensive survey on this topic and posed a number of interesting open questions that have guided the development of the literature. Nine years later, researchers have achieved breakthrough results answering those questions. Moreover, researchers have proposed novel online matching models, motivated by new scenarios such as ride sharing platforms.

In this paper, we survey these recent advances in online matching. We discuss the breakthrough results based on the classical one-sided vertex arrival model in section 1, and summarize those results in section 1.3. Then in section 2, we explore more recently proposed models that go beyond the one-sided vertex arrival setting.

1 One-Sided Vertex Arrival

Take advertising on the Internet as an example:

> Assume we are running a search engine. Advertisers want to display their ads to users who search for certain keywords. When a user performs a search, the search engine needs to immediately pick an advertiser interested in the search term and show the corresponding ad to the user.

This situation can be modeled as an online bipartite matching problem. Consider an underlying bipartite graph with vertices corresponding to the advertisers and the searches. The vertices corresponding to the advertisers are known upfront to the search engine and called *offline vertices*. The vertices corresponding to the searches are called *online vertices*, and they arrive online in a sequence. Upon the arrival of each online vertex, its incident edges to the offline vertices are revealed. In this example, an edge between an online vertex and an offline vertex means that the keyword searched by the user meets the interest of the advertiser. The algorithm then immediately decides which edge to match.

To simplify our explanations, we will introduce some notations but not provide any analysis or proof throughout the survey. However, we will describe and provide insights into some elegant algorithms.

In 1990, Karp et al. [36] first formalized the online bipartite matching problem in the unweighted version to maximize the size of the matching selected by the algorithm.

Online Bipartite Matching (Unweighted)

We use $G = (L \cup R, E)$ to denote the underlying bipartite graph, where L is the set of offline vertices and R is the set of online vertices. Each vertex can be matched once at most. The offline vertices in L are given upfront, and the online vertices in R arrive in a sequence. At the arrival of each online vertex, we reveal its incident edges to offline vertices. We then need to decide whether to match these edges immediately and irrevocably. Our objective is to maximize the cardinality of the matched edges.

Competitive Ratio: An algorithm's competitive ratio Γ is defined as the minimum ratio of the algorithm's (expected) matching size to the maximum matching size in hindsight, over all possible graphs and arrival orders of online vertices. Formally, given an underlying graph $G = (L \cup R, E)$ and an arriving order $\sigma(R)$ of online vertices, we should have the algorithm's solution: $\mathrm{ALG}(G, \sigma(R))$ is at least Γ times the optimal offline maximum matching $\mathrm{OPT}(G)$ on G.

Competitive Ratio (Worst Graph and Worst Order)

- $\Gamma = \min\limits_{G,\sigma(R)} \dfrac{\mathrm{ALG}(G, \sigma(R))}{\mathrm{OPT}(G)}$
- $\Gamma = \min\limits_{G,\sigma(R)} \dfrac{\mathbb{E}[\mathrm{ALG}(G, \sigma(R))]}{\mathrm{OPT}(G)}$ (for randomized algorithms)

1/2 Barrier: A greedy algorithm that simply matches a vertex to an arbitrary unmatched neighbor is 1/2-competitive because it always produces a maximal matching, which is at least half the size of a maximum matching. Moreover, we can prove that it is the best algorithm among all deterministic algorithms.

This observation generalizes to the weighted and arrival variants we introduce later. Therefore, the most natural and fundamental research question is as follows:

How can we design* randomized *online algorithms that give a competitive ratio better than 1/2?

Karp, Vazirani, and Vazirani proposed the celebrated Ranking algorithm that achieves the optimal competitive ratio of $1 - 1/e$.

Ranking Algorithm

Ranking uniformly picks a random permutation over offline vertices at the beginning and then, upon the arrival of each online vertex, matches it to the first unmatched neighbor (if one exists) according to the permutation.

Theorem 1.1 *[36] Ranking achieves a competitive ratio of $1 - 1/e$ for the unweighted online bipartite matching problem.*

Theorem 1.2 *[36] No randomized algorithms can achieve a competitive ratio larger than $1 - 1/e$ for the unweighted online bipartite matching problem.*

1.1 Weighted Models

There are multiple weighted variants of the classical unweighted setting, each of which we describe in the following sections.

Vertex-weighted: In the vertex-weighted version, each offline vertex u is associated with a weight $w(u)$. The goal is to maximize the total weight of matched offline vertices. Aggarwal et al. [1] generalized the Ranking algorithm and attained the same optimal competitive ratio of $1 - 1/e$.

Edge-weighted: In the edge-weighted version, edges can have different weights. The goal is to maximize the total weight of the selected matching. It is found that this model does not admit any non-trivial result, even in the case of a single offline vertex. A common assumption is the free disposal assumption [16, 39]. It is allowed to dispose of previously matched edges to match new edges with larger weights. In 2020, Fahrbach et al. [15] achieves a 0.5086 competitive ratio. This breakthrough is the first time that the long-standing 1/2 barrier has been overcome. Follow-up works [4, 22] have subsequently improved the ratio to 0.536.

AdWords: In the AdWords problem, each offline vertex u is associated with a budget B_u, and the edges are associated with a bid b_{uv}. The reward for matching it to a set of online vertices S equals the minimum between its budget and the total weight (bid) of its incident edges to S; specifically, $r_u = \min\left\{B_u, \sum_{v\in S} b_{uv}\right\}$. Therefore, this is also referred to as a budget-additive setting. A well-known assumption under this model is the small bid assumption, which assumes each edge's bid b_{uv} is very small compared to the corresponding B_u. With this assumption, the MSVV algorithm proposed by [47] achieves the optimal competitive ratio of $1-1/e$. However, if we allow general bids, no algorithm could beat 0.5 until 2020. Huang, Zhang, and Zhang [31] gave a 0.5016-competitive algorithm, breaking the trivial 0.5 bound for the first time.

Submodular Welfare: The submodular welfare maximization model is the most generalized. In this model, each offline vertex u is associated with a valuation function f_u. Algorithms can allocate online vertices to offline vertices at their arrival time irrevocably. Each offline vertex u earns the revenue $f_u(V_u)$, where V_u is the set of online vertices allocated to it. The goal is to maximize the total welfare of offline vertices, specifically, $\sum_{u\in L} f(V_u)$. We can check that all the models described above are special cases of this generalized model (if we assume free disposal in the edge-weighted model). Lehmann et al. [42] proved that a greedy algorithm can achieve 1/2 in this model, which is already optimal even among randomized algorithms.

Negative results: The vertex-weighted variant, edge-weighted (free disposal) variant, AdWords (general bids) problem, and submodular welfare problem share the same worst case as the unweighted version ($1-1/e$ by Karp et al. [36]) because they all generalize the unweighted model. We remark that it is the best known upper bound for them even with unlimited computation power. For the AdWords problem with the small bid assumption, although it is not directly generalized by the unweighted problem, Mehta et al. [47] proved that no randomized algorithm can beat $1-1/e$ in this model. If we restrict the scope to polynomial time algorithms, no randomized algorithm can beat 1/2 in the submodular welfare problem unless NP = RP.

1.2 Stochastic Arrival Models

Besides the weighted generalizations of the classical model, researchers have attempted to overcome the $1-1/e$ barrier by relaxing the worst-case arrival order. Specifically, they apply the random arrival model and the i.i.d. arrival model with redefined competitive ratios. Both models have weighted variants, as described in the previous section. It is worth noting that the random arrival model is strictly harder than the i.i.d. arrival model. That is, if we consider the same weighted variant in those two models and the same algorithm, the competitive ratio defined in the random arrival model is always at most the competitive ratio in the i.i.d. model. We prove this observation after illustrating the definitions of competitive ratios in the two arrival models.

Random Arrival Model

In the random arrival model, there is a uniformly random permutation $\pi(R)$ of online vertices R, and the algorithm is qualified by its expected performance over the randomness. The competitive ratio is defined as the minimum ratio of the algorithm's expected solution to the offline optimal solution over all possible underlying graphs.

Competitive Ratio for the Random Arrival Model (Worst Graph and Random Arrival)

$$\Gamma = \min_{G} \frac{\mathbb{E}[\mathrm{ALG}(G, \pi(R))]}{\mathrm{OPT}(G)}$$

Results: If we apply the random arrival model to the unweighted online bipartite matching problem, Mahdian and Yan prove that Ranking can achieve a competitive ratio of 0.696 [43] (and 0.656 by Karande et al. [35]). Those works were the first two to beat $1-1/e$ (achieved by a greedy algorithm) results in this model proposed in 2011. For the upper bound, Karande et al. provide an example that Ranking achieves a ratio of 0.726; Mahdian and Yan show that no (randomized) algorithm can beat 0.83 even in the known i.i.d. model. Since the known i.i.d. model is strictly easier than the random arrival model, it also implies an upper bound in the random arrival model.

In the vertex-weighted version, Huang et al.'s result in 2018 [28] was the first to beat $1-1/e$, proving that the weighted Ranking is 0.653-competitive. Jin and Williamson [33] subsequently improved the result to 0.662.

For the AdWords problem with the small bid assumption, a greedy algorithm can also achieve $1-1/e$ by Goel and Mehta [23]. In 2009, Devanur and Hayes [9] gave the near-optimal $(1-\epsilon)$-competitive algorithm, and Mirrokni et al. proved a competitive ratio of 0.76 for the MSVV algorithm. Note that MSVV achieves a competitive ratio of $1-1/e$ in the worst-case model with the small bid assumption. If we allow general bids in the AdWords problem, it is still open for improvements over the stronger worst-case model, or improvements over the generalized submodular welfare model (also random arrival model).

In the submodular welfare model, Korula et al. [40] first beat 0.5 under the random arrival assumption. This result has subsequently been improved to 0.5096 by Buchbinder et al. [7].

i.i.d. Arrival Model

In the i.i.d. arrival model, algorithms recognize that the online vertices will be drawn from a given distribution of several possible types, and the type encodes the vertex's incident edges and weights (if they exist). All the arriving online vertices are independently drawn from this given distribution. Because the graph G in this model is itself randomized, $\mathrm{OPT}(G)$ is correspondingly random. The definition of competitive ratio becomes the minimum ratio of the expectation of $\mathrm{ALG}(G)$ (over the randomness of the algorithm and the graph) to the expectation of $\mathrm{OPT}(G)$ (over the randomness of G).

Competitive Ratio for i.i.d. Arrival
(Worst Graph and Worst Distribution)

$$\Gamma = \min_{D} \frac{\mathbb{E}[\mathrm{ALG}(G)]}{\mathbb{E}[\mathrm{OPT}(G)]}$$

Why is the i.i.d. arrival model easier than the random arrival model? There are two specific settings in the i.i.d. arrival model: known i.i.d. setting and unknown i.i.d. setting. They differ in whether the distribution is given to the algorithm upfront. To prove that the i.i.d. model is easier than the random arrival model, we assume the following: If we have an algorithm that is at least Γ-competitive in the random arrival model, then it is a Γ-competitive algorithm in the i.i.d. model, even without any knowledge of the distribution. The events in the probability space of the i.i.d. model can be viewed as a random graph plus a random arrival model of online vertices. Considering each possible graph, the expected solution of a Γ-competitive algorithm in the random arrival model is better than Γ times OPT. Consequently, its expected solution is at least Γ times the expected OPT over the randomness of graphs.

Results: In the unweighted version, Feldman et al. [17] under the integral rate assumption was the first to beat $1-1/e$. Manshadi et al. [44] provided the first work without this assumption, and Jaillet and Lu [32] subsequently improved the result to 0.706 (which was state-of-the-art before 2013). Then in 2021, Huang and Shu gave the state-of-the-art 0.711-competitive algorithm.

For the vertex-weighted variant, Huang and Shu [26] improved the 0.662-competitive algorithm in the random arrival model to a 0.7009-competitive algorithm.

For the edge-weighted version with free disposal, we can achieve $1-1/e$ in both unknown i.i.d. and known i.i.d. models. This is implied from the $(1-1/e)$-competitive algorithm in the unknown i.i.d. model for the submodular welfare problem.

For the AdWords problem with the small bid assumption, Devanur et al. [10] gave a better near-optimal algorithm. Although this algorithm and the near-optimal algorithm [11] both achieve a competitive ratio of $1-\epsilon$, the ϵ parameter in this algorithm becomes better in the unknown i.i.d. model and even better in the known i.i.d. model. If we allow general bids, we know that a greedy algorithm achieves a competitive ratio of $1-1/e$ in the unknown i.i.d. model, as proved by Devanur et al. [10].

For the submodular welfare problem, Kapralov et al. [34] illustrated a $(1-1/e)$-competitive algorithm in the unknown i.i.d. model. This is also the state-of-the-art result for the known i.i.d. model.

Negative Results: These stochastic models help us to overcome the hardness barrier in the worst-case analysis. However, what is the best result that we can obtain? Manshadi et al. [44] proved that no randomized algorithm can beat 0.823 in the unweighted model with known i.i.d. arrival setting. Note that known i.i.d. is the easiest stochastic model of the three, so 0.823 can serve as the upper bound for unweighted, vertex-weighted, edge-weighted, AdWords (general bids), and submodular welfare generalization under the random, unknown i.i.d., and known i.i.d. arrival models.

For the AdWords problem with the small bid assumption, [10] proved that no randomized algorithm can achieve

$1-o(\sqrt{\gamma})$, where γ is the maximum bid-to-budget ratio. (Here, γ is small enough with the small bid assumption.)

If we restrict the computational power, Kapralov et al. [34] proved that no randomized algorithm can beat $1-1/e$ in the submodular welfare model under unknown i.i.d. arrival, unless NP = RP.

1.3 Results Conclusion

The following two tables summarize the results of different variants of the one-sided vertex arrival model.

In Table 1, **Bold** indicates the advanced results after 2013, and arrows indicate that the ratio is derived by a stronger result. For example, the left arrow following the ratio 0.696 in the unweighted unknown i.i.d. model indicates that the ratio comes from the 0.696 competitive ratio in the unweighted random arrival model (harder).

In Table 2, we list the hardness results. We use a star $*$ to indicate that the result needs a computational assumption. The γ parameter denotes the bid-to-budget ratio in the AdWords problem.

1.4 Other Variants

Although the online matching problem is essentially driven by online advertisement, the presented variants are unable to precisely capture all real-world scenarios. That is why people still try to formulate different models to capture various aspects of real-world applications. In this section, we list some associated topics.

Studying online matching problems on restricted graph instances raises interesting research questions. There is a line of work considering graphs with bounded degrees. Buchbinder et al. [5] showed that we can achieve a competitive ratio of $1-(1-1/d)^d$ by a *deterministic* algorithm if the degree of each online vertex is at most d. Azar et al. [3] proved the ratio is optimal even among

Table 1 State-of-the-art positive results for one-sided vertex arrival models

	Worst Case	Random Arrival	Unknown i.i.d.	Known i.i.d.
Unweighted	$1-1/e$([36])	0.696([43])	0.696($\leftarrow$)	**0.711**([26])
Vertex-weighted	$1-1/e$([1])	**0.662**([33])	0.662($\leftarrow$)	**0.701**([26])
AdWords (general bids)	**0.5016**([31])	0.5096($\downarrow$)	$1-1/e$([10])	$1-1/e$($\leftarrow$)
Submodular welfare	1/2	**0.5096**([40])	$1-1/e$([34])	$1-1/e$($\leftarrow$)
Edge-weighted (free disposal)	**0.536**([4, 22])	0.536($\leftarrow$)	$1-1/e$($\uparrow$)	$1-1/e$($\uparrow$)
AdWords (small bids)	$1-1/e$([47])	$1-\epsilon$([11])	$1-\epsilon$([10])	$1-\epsilon$([10])

Table 2 State-of-the-art negative results for one-sided vertex arrival models

	Worst Case	Random Arrival	Unknown i.i.d.	Known i.i.d.
Unweighted	$1-1/e$([36])	0.823($\rightarrow$)	0.823($\rightarrow$)	0.823([44])
Vertex-weighted	$1-1/e$($\uparrow$)	0.823($\rightarrow$)	0.823($\rightarrow$)	0.823($\uparrow$)
Edge-weighted (free disposal)	$1-1/e$($\uparrow$)	0.823($\rightarrow$)	0.823($\rightarrow$)	0.823($\uparrow$)
AdWords (small bids)	$1-1/e$([47])	$1-o(\sqrt{\gamma})$($\rightarrow$)	$1-o(\sqrt{\gamma})$($\rightarrow$)	$1-o(\sqrt{\gamma})$([10])
AdWords (general bids)	$1-1/e$($\uparrow$)	0.823($\rightarrow$)	0.823($\rightarrow$)	0.823($\uparrow$)
Submodular welfare	1/2($*$, [34])	$1-1/e$($*$, $\rightarrow$)	$1-1/e$($*$, [34])	0.823($\uparrow$)

randomized algorithms. If we additionally restrict the degrees of offline vertices to at least k, Naor and Wajc [49] proved a competitive ratio of $1-(1-1/d)^k$ by a *deterministic* algorithm. The result implies that we can achieve a competitive ratio of $1-(1-1/d)^d$ by a deterministic algorithm on d-regular graphs. Focusing on d-regular graphs, Cohen and Wajc [8] subsequently proposed a randomized algorithm that achieves a near-optimal competitive ratio of $1-O(\frac{\sqrt{\log d}}{\sqrt{d}})$.

Mehta and Panigrahi [46] formulated a variant called stochastic rewards. In the classical online bipartite matching model, when we decide to match an edge, we earn one reward, and the matched online and offline vertices are eliminated. However, in real-world scenarios, even if we display the advertisement to an online user, the user will click the advertisement with a predicted click-through-rate. On the other hand, although the online user is gone, we can match the offline unclicked advertisement again in the future. To this end, the stochastic reward model associates each edge with a success probability. We can determine whether the user (online vertex) clicks the displayed advertisement after we decide to match them irrevocably. The objective is to maximize the number of offline vertices that are successfully matched. Mehta and Panigrahi [46] first studied this model with the special case of equal probability and proposed a 0.534-competitive algorithm, and an upper bound of 0.621. They also studied the special case of vanishing probabilities (i.e., all success probabilities tend to zero) and showed a competitive ratio of 0.567 in the case of both equal and vanishing probabilities. Later, Huang and Zhang [27] improved the ratio to 0.576. Mehta et al. [48] proposed a 0.534-competitive algorithm that only needs vanishing probabilities. Huang and Zhang [27] subsequently improved this to 0.572.

Feng et al. [20] initiated the study of the two-stage stochastic online bipartite matching problem. In this model, one side of the vertices is divided into two parts: D_1 and D_2. In the first stage, D_1 is revealed to the algorithm, which makes matching decisions for this set of vertices. In the second stage, D_2 is drawn stochastically according to a prior known distribution, and the algorithm makes decisions based on the realization of the graph. They achieved a tight 3/4 competitive ratio for this problem.

Feng and Niazadeh [19] studied the K-stage variants of the classical vertex weighted bipartite matching and AdWords problems, where online vertices arrive in K batches. They designed optimal $(1-(1-1/K)^K)$-competitive fractional matching algorithms for both settings.

2 Beyond One-Sided Arrival

2.1 Generalized Arrival Models

Besides the weighted generalization, another interesting topic is generalizing one-sided online bipartite matching via the arrival pattern. The one-sided online model assumes that the underlying graph is bipartite and one-sided vertices are given upfront. Naturally then, one question this raises is:

What if we allow all vertices to arrive online?

General Vertex Arrival: Wang and Wong [51] first formulated the general vertex arrival model in 2015. Let $G=(V,E)$ be the underlying graph and *all vertices* in V arrive in an online fashion. At the arrival of each vertex, we reveal its incident edges with vertices that have already arrived. We can match v to its arrived unmatched neighbors at v's arrival time, or we can leave it unmatched for the future. Because we relax the restriction that one-sided vertices should be offline, defining problems on *general graphs* becomes natural.

Wang and Wong [51] studied the fractional online algorithm in the model and proposed a 0.526-competitive dual-based algorithm. In 2019, Gamlath et al. [21] first beat the 0.5 barrier by an integral algorithm with a small constant ϵ. On the negative side, the state-of-the-art result by Buchbinder et al. [6] shows that even fractional algorithms (stronger than any randomized integral algorithm) cannot achieve a competitive ratio of 0.591. (Very recently, however, this has been improved to 0.583.)

Next, we consider online algorithms. Note that we only allow algorithms to reveal and make matching decisions for edges (u,v) at the later arrival time of u and v. So then, why can algorithms not make a better choice after revealing more edges? In the following model, we allow algorithms to match edges after the time we reveal them, but at most when one of u and v leaves.

Fully Online (Vertices with Deadline): Huang el al. [29] first proposed the fully online matching model. Let $G=(V,E)$ be the underlying graph, *all vertices* in V arrive in an online fashion, and each edge (u,v) can be revealed after both u and v arrive. Each vertex corresponds to a deadline. We can match vertices any time before their deadline and assume that there is no edge between two vertices such that one arrives later than the other's deadline.

Huang et al. [29] first studied the well-behaved Ranking

algorithm in the fully online model and proved that Ranking is 0.567-competitive (tight for Ranking) if the underlying graph is bipartite and 0.5211-competitive for general cases. Later, Huang et al. [30] gave the balanced Ranking algorithm that improved the ratio to 0.569, and subsequently provided a fractional algorithm that achieves 0.592-competitive (which has very recently been improved to 0.6). On the negative side, Eckl et al. [12] proved that no fractional algorithm can beat the ratio of 0.629.

We can further generalize the arrival patterns.

Edge Arrival: Gamlath et al. [21] studied the most generalized and hardest arrival model, namely, the edge arrival model. Let $G = (V, E)$ be the underlying graph and *all edges* in E arrive in an online fashion. At the arrival of an edge, we need to decide whether to match it (if we can) immediately.

Unfortunately, Gamlath et al. [21] gave a very strong negative result that no fractional algorithm can beat 0.5 in the model. This means that the trivial greedy algorithm is already optimal.

Conclusion and Comparison: Table 3 provides a summary of the results. The three models can be ordered by difficulty: Fully online $\leq_{easier}$ General vertex arrival $\leq_{easier}$ Edge arrival. The results in the table have already shown some separation among these three models. A fractional algorithm in the fully online model beat the hardness barrier in the general vertex arrival model, indicating that the fully online model is strictly "easier" than the general vertex arrival, at least in terms of fractional algorithms. Moreover, the edge arrival model has a strong hardness result of 1/2, meaning that it is strictly harder than the other two models.

2.2 Weighted Models

Next, we move to the edge-weighted variants of the different arrival models mentioned earlier. Note that no non-trivial theoretical results can be achieved without making extra assumptions. Even in the classical one-sided arrival model with only one offline vertex, no constant competitive algorithm can be achieved if weighted edges are considered.

The two most popular models studied in the literature assume the random arrival order of online vertices or the stochastic information of online vertices. The two models are also known as *secretary matching* setting and *prophet matching* setting. The secretary setting generalizes the classical secretary problem, and the prophet setting generalizes the prophet inequality. In the prophet setting, we assume online vertices are drawn from known distributions independently, while the distributions are not necessarily identical. To remain consistent with previous sections, we refer to the setting as stochastic non-i.i.d. setting. Note that the secretary and prophet settings are not comparable in general.

In addition to the secretary and prophet settings, the online windowed matching model by Ashlagi et al. [2] is a simultaneous and similar work to the fully online matching model by [29] that applies to edge-weighted graphs. To enable non-trivial theoretical results, the online windowed matching model makes extra assumptions.

We summarize all known results in Table 4 and describe them later.

One-Sided Vertex Arrival: For the classical one-sided arrival model, tight competitive algorithms have been

Table 3 State-of-the-art results for different arrival models in worst cases

	Integral (Randomized)	Fractional	Hardness
One-sided	$1 - 1/e$	$1 - 1/e$	$1 - 1/e$
Fully online (bipartite)	0.569	0.6	0.613
Fully online (general)	0.5211	0.6	0.613
General vertex arrival	1/2 + ϵ	0.526	0.584
Edge arrival	1/2	1/2	1/2

Table 4 State-of-the-art results for the edge-weighted variants of different arrival models

Edge-Weighted	Worst Case	Random Arrival	Stochastic Non-i.i.d.
One-sided (no disposal)	-	$1/e$	1/2
General vertex arrival	-	5/12	1/2
Edge arrival	-	1/4	0.337
Online windowed	1/4	0.279	-

designed in the two models. Kesselheim et al. [37] generalized the classical $1/e$-competitive algorithm for the secretary problem to the one-sided online bipartite matching setting. For the prophet matching setting, Feldman et al. [18] generalized the classical 1/2-competitive algorithm for the prophet inequality to the one-sided online bipartite matching setting. The two bounds are both optimal.

General Vertex Arrival: Ezra et al. [13, 14] studied the secretary matching and prophet matching under the general vertex arrival model and achieved tight competitive ratios of 5/12 and 1/2. Interestingly, although the general vertex arrival model is a generalization of the one-sided arrival model in the worst-case arrival order, the hardness of $1/e$ under the random arrival assumption does not carry over. The result of Ezra et al. [13] shows that it is indeed easier to match when all vertices arrive online in a random order.

Edge Arrival: The secretary matching and prophet matching settings under edge arrival are studied more extensively in the algorithmic game theory literature. The state-of-the-art algorithmic results are 1/4 and 0.337 competitive ratios by Ezra et al. [13, 14]. The best known hardness results are an upper bound of $1/e$ for the secretary matching, inherited from the classical secretary problem, and an upper bound of 3/7 by Pollner [50]. Prior to that, the prophet setting was studied by [24] for bipartite graphs and by Kleinberg and Weinberg [38] in the constraint of matroid intersection, which subsumes bipartite prophet matching as a special case. For secretary, Kesselheim et al. [37] established a competitive ratio of $1/2e$, by a reduction from edge arrival to vertex arrival in hypergraphs.

Edge-Weighted Online Windowed Matching: Simultaneous to the work of Huang et al. [29], Ashlagi et al. [2] introduced the online windowed matching model. This model is similar to the fully online matching model with an extra assumption of *first-in, first-out*, meaning that a vertex with an earlier arrival time would also have an earlier departure time. With this extra assumption, Ashlagi et al. [2] provided a 1/4-competitive algorithm for the edge-weighted graphs and a 0.279-competitive algorithm under the random arrival model. On the negative side, they showed an upper bound of 1/2.

2.3 Other Variants

Very recently, Gravin et al. [25] studied unweighted stochastic matching with edge arrival on bipartite graphs and achieved a 0.503-competitive algorithm, beating the 0.5-barrier in the worst-case setting.

Lee and Singla [41] proposed the batched edge arrival model in online matching. That is, the edges of the graph are shown in batches at each time step. When the graph is revealed in 2 stages, they designed a 2/3-competitive algorithm. When the graph is shown in s stages, they presented a randomized algorithm with a competitive ratio of $\frac{1}{2} + 2^{-O(s)}$.

References

[1] G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *SODA*, pages 1253–1264, 2011.

[2] I. Ashlagi, M. Burq, C. Dutta, P. Jaillet, A. Saberi, and C. Sholley. Edge weighted online windowed matching. In *EC*, pages 729–742, 2019.

[3] Y. Azar, I. R. Cohen, and A. Roytman. Online lower bounds via duality. In *SODA*, pages 1038–1050. SIAM, 2017.

[4] G. Blanc and M. Charikar. Multiway online correlated selection. *CoRR*, abs/2106.05579, 2021.

[5] N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, volume 4698 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2007.

[6] N. Buchbinder, D. Segev, and Y. Tkach. Online algorithms for maximum cardinality matching with edge arrivals. *Algorithmica*, 81(5):1781–1799, 2019. doi: 10.1007/s00453-018-0505-7. URL https://doi.org/10.1007/s00453-018-0505-7.

[7] N. Buchbinder, M. Feldman, Y. Filmus, and M. Garg. Online submodular maximization: beating 1/2 made simple. *Math. Program.*, 183(1):149–169, 2020. doi: 10.1007/s10107-019-01459-z. URL https://doi.org/10.1007/s10107-019-01459-z.

[8] I. R. Cohen and D. Wajc. Randomized online matching in regular graphs. In A. Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 960–979. SIAM, 2018. doi: 10.1137/1.9781611975031.62. URL https://doi.org/10.1137/1.9781611975031.62.

[9] N. R. Devanur and T. P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In J. Chuang, L. Fortnow, and P. Pu, editors, *Proceedings 10th ACM Conference on Electronic Commerce (EC-2009), Stanford, California, USA, July 6–10, 2009*, pages 71–78. ACM, 2009. doi: 10.1145/1566374.1566384. URL https://doi.org/10.1145/1566374.1566384.

[10] N. R. Devanur, K. Jain, B. Sivan, and C. A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *J. ACM*, 66(1):7:1–7:41, 2019. doi: 10.1145/3284177. URL https://doi.org/10.1145/3284177.

[11] N. R. Devenur and T. P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *ACM Conference on Electronic Commerce*, pages 71– 78, 2009.

[12] A. Eckl, A. Kirschbaum, M. Leichter, and K. Schewior. A stronger impossibility for fully online matching. *Operations Research Letters,* to appear.

[13] T. Ezra, M. Feldman, N. Gravin, and Z. G. Tang. Secretary matching with general arrivals. *CoRR*, abs/2011.01559, 2020.

[14] T. Ezra, M. Feldman, N. Gravin, and Z. G. Tang. Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models. In *EC*, pages 769–787. ACM, 2020.

[15] M. Fahrbach, Z. Huang, R. Tao, and M. Zadimoghaddam. Edge-weighted online bipartite matching. In *FOCS*, pages 412–423. IEEE, 2020.

[16] J. Feldman, N. Korula, V. S. Mirrokni, S. Muthukrishnan, and M. Pál. Online ad assignment with free disposal. In *WINE*, pages 374–385, 2009.

[17] J. Feldman, A. Mehta, V. S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 117–126. IEEE Computer Society, 2009. doi: 10.1109/FOCS.2009.72. URL https://doi.org/10.1109/FOCS.2009.72.

[18] M. Feldman, N. Gravin, and B. Lucier. Combinatorial auctions via posted prices. In *SODA*, pages 123–135. SIAM, 2015.

[19] Y. Feng and R. Niazadeh. Batching and optimal multi-stage bipartite allocations (extended abstract). In *ITCS*, volume 185 of *LIPIcs*, pages 88:1–88:1. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[20] Y. Feng, R. Niazadeh, and A. Saberi. Two-stage stochastic matching with application to ride hailing. In *SODA*, pages 2862–2877. SIAM, 2021.

[21] B. Gamlath, M. Kapralov, A. Maggiori, O. Svensson, and D. Wajc. Online matching with general arrivals. In *FOCS*, pages 26–37. IEEE, 2019.

[22] R. Gao, Z. He, Z. Huang, Z. Nie, B. Yuan, and Y. Zhong. Improved online correlated selection. *CoRR*, abs/2106.04224, 2021.

[23] G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, pages 982–991, 2008.

[24] N. Gravin and H. Wang. Prophet inequality for bipartite matching: Merits of being simple and non adaptive. In *EC*, pages 93–109. ACM, 2019.

[25] N. Gravin, Z. G. Tang, and K. Wang. Online stochastic matching with edge arrivals. In *ICALP*, volume 198 of *LIPIcs*, pages 74:1–74:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[26] Z. Huang and X. Shu. Online stochastic matching, Poisson arrivals, and the natural linear program. In *STOC*, pages 682–693. ACM, 2021.

[27] Z. Huang and Q. Zhang. Online primal dual meets online matching with stochastic rewards: configuration LP to the rescue. In *STOC*, pages 1153–1164. ACM, 2020.

[28] Z. Huang, Z. G. Tang, X. Wu, and Y. Zhang. Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. *ACM Transactions on Algorithms*, 15(3):1–15, 2019.

[29] Z. Huang, N. Kang, Z. G. Tang, X. Wu, Y. Zhang, and X. Zhu. Fully online matching. *J. ACM*, 67(3):17:1–17:25, 2020.

[30] Z. Huang, Z. G. Tang, X. Wu, and Y. Zhang. Fully online matching II: beating ranking and water-filling. In *FOCS*, pages 1380–1391. IEEE, 2020.

[31] Z. Huang, Q. Zhang, and Y. Zhang. Adwords in a panorama. In *FOCS*, pages 1416–1426. IEEE, 2020.

[32] P. Jaillet and X. Lu. Online stochastic matching: New algorithms with better bounds. *Math. Oper. Res.*, 39(3):624–646, 2014. doi: 10.1287/moor.2013.0621. URL https://doi.org/10. 1287/moor.2013.0621.

[33] B. Jin and D. P. Williamson. Improved analysis of RANKING for online vertex-weighted bipartite matching. *CoRR*, abs/2007.12823, 2020.

[34] M. Kapralov, I. Post, and J. Vondrák. Online submodular welfare maximization: Greedy is optimal. In S. Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1216–1225. SIAM, 2013. doi: 10.1137/1.9781611973105.88. URL https://doi.org/10.1137/1.9781611973105.88.

[35] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.

[36] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.

[37] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *ESA*, volume 8125 of *Lecture Notes in Computer Science*, pages 589–600. Springer, 2013.

[38] R. Kleinberg and S. M. Weinberg. Matroid prophet inequalities and applications to multi-dimensional mechanism design. *Games Econ. Behav.*, 113:97–115, 2019.

[39] N. Korula, V. S. Mirrokni, and M. Zadimoghaddam. Bicriteria online matching: Maximizing weight and cardinality. In *WINE*, volume 8289 of *Lecture Notes in Computer Science*, pages 305–318. Springer, 2013.

[40] N. Korula, V. S. Mirrokni, and M. Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM J. Comput.*, 47(3):1056–1086, 2018. doi: 10.1137/ 15M1051142. URL https://doi.org/10.1137/15M1051142.

[41] E. Lee and S. Singla. Maximum matching in the online batch-arrival model. *ACM Trans. Algorithms*, 16(4):49:1–49:31, 2020.

[42] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games Econ. Behav.*, 55(2):270–296, 2006. doi: 10.1016/ j.geb.2005.02.006. URL https://doi.org/10.1016/ j.geb.2005.02.006.

[43] M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *STOC*, pages 597–606, 2011.

[44] V. H. Manshadi, S. O. Gharan, and A. Saberi. Online stochastic matching: Online actions based on offline statistics. *Math. Oper. Res.*, 37(4):559–573, 2012. doi: 10.1287/moor.1120. 0551. URL https://doi.org/10.1287/ moor.1120.0551.

[45] A. Mehta. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, 8(4): 265–368, 2013.

[46] A. Mehta and D. Panigrahi. Online matching with stochastic rewards. In *FOCS*, pages 728–737. IEEE, 2012.

[47] A. Mehta, A. Saberi, U. V. Vazirani, and V. V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

[48] A. Mehta, B. Waggoner, and M. Zadimoghaddam. Online stochastic matching with un- equal probabilities. In P. Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1388–1404. SIAM, 2015. doi: 10.1137/1.9781611973730.92. URL https://doi.org/10.1137/1.9781611973730.92.

[49] J. S. Naor and D. Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Trans. Economics and Comput.*, 6(3-4):16:1–16:20, 2018. doi: 10.1145/3105447. URL https://doi.org/10.1145/3105447.

[50] T. Pollner. Two problems in combinatorial optimization under uncertainty, 2020.

[51] Y. Wang and S. C. Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *ICALP*, pages 1070–1081, 2015.

Learning Augmented Algorithm Design for Combinatorial Optimization

Lingxiao Huang, Yuyi Wang, Xiang Yan
TCS Lab

Abstract

Combinatorial optimization is one of the most important areas in computer science and is applied in various heterogeneous application domains. Traditional research mainly focuses on solving combinatorial optimization problems via theoretical or heuristic algorithms. Recently, there have been some initial studies on incorporating machine learning (ML) to help solve combinatorial optimization problems. In this proposal, we first survey and introduce the application of three paradigms to combinatorial optimization, including theoretical, heuristic, and ML algorithms, as well as their combinations. Then we summarize some limitations and challenges in current uses of ML algorithms and propose potential approaches and solutions to overcome them. Our goal is to design unified frameworks that apply machine learning to augment traditional theoretical or heuristic algorithms for combinatorial optimization, both in theory and in application scenarios.

1 Problem Formulation

Combinatorial optimization is one of the most important topics in computer science. It has a wide range of application in many industries, including electronic design automation (EDA), cloud computing, autonomous driving, compiler, database, supply chain, and finance. In general, combinatorial optimization can be viewed as finding an optimal object from a finite set of objects. Combinatorial optimization problems are usually difficult to solve because exhaustive search methods are not tractable in many such problems (i.e., NP problems). A large number of researches on combinatorial optimization have been carried out in the fields of operations research, algorithm theory, and computational complexity theory. Typical problems include the traveling salesman problem (TSP), resource management, routing, job scheduling, bin packing, and expression simplification, as shown in Figure 1.

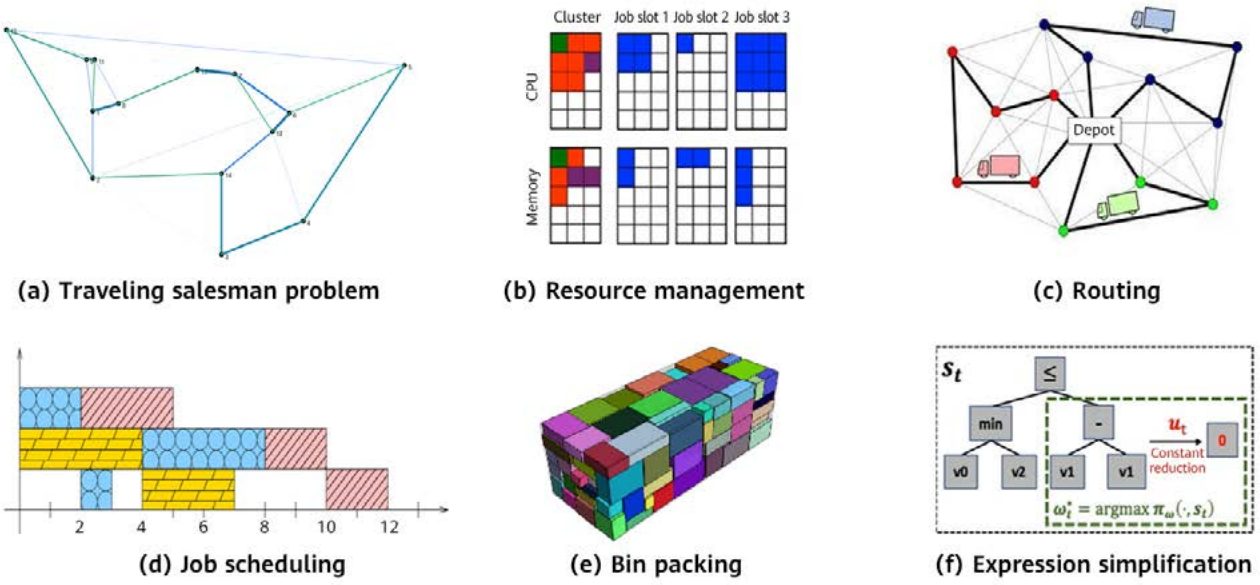
Traditional solutions for combinatorial optimization are mainly theoretical algorithms or heuristic algorithms such as greedy, mixed-integer linear programming (MILP), and local search. The output solution may not be an optimal one but usually has a provable guarantee on either the solution quality or the execution time. Due to their importance, combinatorial optimization problems attract the attention of major companies, and a lot of traditional solution-based software has been developed, including OR-Tools (Google), CPLEX (IBM), MindOpt (Alibaba), Gurobi, and Z3 (Microsoft).

Recently, with the rapid development of machine learning (ML), there are studies on applying ML models to combinatorial optimization in order to improve solution quality. A well-known example is AlphaGo (Google), which effectively combines heuristic algorithms (Monte Carlo tree search) and ML models (deep reinforcement learning). Other examples include Google's AlphaStar and AlphaFold (Google), as well as Facebook's ELF OpenGo and ReLA. Alibaba also incorporates ML models to improve its packing strategies [34].

However, the study of augmenting theoretical/heuristic algorithms by using ML is still in its initial stages. Also, due to specific properties of combinatorial optimization, there are several limitations and challenges that need to be addressed by incorporating ML, e.g., sensitivity, scalability, and adaptability. We aim to investigate more combination approaches between theoretical/heuristic and ML algorithms, and design unified augmented algorithmic frameworks that address existing limitations.

2 Background

Generally speaking, a combinatorial optimization problem has the following form: given a ground set $\mathcal{E} = \{1, 2, \ldots, n\}$, a feasible region $\mathcal{X}$, and an objective function $f : 2^{\mathcal{E}} \to R$, a minimization (resp. maximization) combinatorial optimization problem searches for an optimal solution $x^* \in \mathcal{X}$ such that $f(x^*) \leq f(x)$ (resp. $f(x^*) \geq f(x)$) for all $x \in \mathcal{X}$.



(a) Traveling salesman problem (b) Resource management (c) Routing

(d) Job scheduling (e) Bin packing (f) Expression simplification

Figure 1 Classic combinatorial optimization problems

In this section, we first introduce some important combinatorial optimization problems. Then we introduce three paradigms of algorithmic design for combinatorial optimization, including theoretical, heuristic, and ML algorithms, and analyze their advantages and disadvantages. Next, we introduce existing combinations between these paradigms. Finally, we list classic theoretical and heuristic approaches and review their current combinations with ML.

2.1 Combinatorial Optimization Problems

There are various types of combinatorial optimization problems. We roughly divide combinatorial optimization problems into three types: combinatorial optimization problems on graphs, scheduling and management, and logic problems. For each type, we list some examples that have major applications in Huawei.

Combinatorial Optimization Problems on Graphs

- **Vertex cover.** Given a graph $G = (V, E)$ where V is the vertex set and E is the edge set, the vertex cover problem is to find a set $S \subseteq V$ with minimum cardinality $|S|$ such that S includes at least one vertex of each edge $e \in E$. Some applications are as follows:
  - Wireless - wireless base-station location arrangement
- **Maxcut.** Let $G = (V, E)$ be a graph with vertices V and edges E. A cut of the graph G is a partition of the vertices of G into two disjoint subsets $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ with $V_1 \cup V_2 = \emptyset$. The size of a cut is the number of edges crossing the cut. The maximum cut is defined to be the cut of a graph G whose size is at least as large as any other cut. Some applications are as follows:
  - EDA - graph partition
- **TSP.** Given a finite set of cities N and a distance matrix $(c_{ij})(i, j \in N)$, determine $\min_\pi \sum_{i \in N} c_{i\pi(i)}$, where π runs over all cyclic permutations of N, and specifically, $\pi(i)$ denotes the next city reached by salesman from city i along π. Some applications are as follows:
  - EDA - drilling of printed circuit boards, chip insertion problems
  - Supply chain - the order-picking problem in warehouses
  - Computer network wiring
- **Clustering.** Graphs are structures formed by a set of vertices (also called nodes) and a set of edges that are connections between pairs of vertices. Graph clustering is the task of grouping the vertices of the graph into clusters taking into consideration the edge structure of the graph in such a way that there should be many edges within each cluster and relatively few between the clusters. Some applications are as follows:
  - Cloud computing - data analysis
  - Huawei Mobile Services (HMS) advertising - community discovery
  - Database - quick index

Scheduling and Management

- **Makespan.** Given a list of n tasks where each task i has an execution time t_i, and m parallel machines, the makespan problem is to construct an allocation scheme $\pi : [n] \to [m]$ such that the makespan time $\max_{j \in [m]} \sum_{i \in [n]: \pi(i) = j} t_i$ is minimized. Some applications are as follows:
  - Compiler - pipeline arrangement
  - Distributed system - scheduling
- **Bin packing.** Given a finite list $L = (a_1, a_2, ..., a_n)$ of real numbers in the range $(0, 1]$, and a sequence of unit-capacity bins, $BIN_1, BIN_2, ..., BIN_m$ extending from left to right. The problem is to find an assignment or packing of the numbers into the bins so that no bin has contents totaling more than one, and yet the number of bins used, i.e., nonempty, is minimized. Some applications are as follows:
  - Supply chain - packing
- **Routing.** In Q-routing, the routing decision-maker at each node x makes use of a table of values $Q_x(y, d)$, where each value is an estimate, for a neighbor y and destination d, of how long it takes for a packet to be delivered to node d, if sent via neighbor y, excluding the time spent in node x's queue. When node x has to make a routing decision it simply chooses the neighbor y for which $Q_x(y, d)$ is minimum, namely $Q_y(z, d) = \min_{\tilde{z} \in N(y)} Q_y(z, d)$. Some applications are as follows:
  - Digital communication - router
  - Supply chain - vehicle scheduling

Logic Problems

- **Constraint satisfaction problem (CSP)**. The problem is expressed by a set of variables and a set of constraints. The variables are the unknown of the problem and each of them has a finite domain of values. A *solution* is a complete assignment of the variables by a tuple of values from their respective domains, and has to respect

some conditions expressed by constraints. A constraint is a relation on some variables and restricts the values that the variables can take. The solution of a CSP is thus an assignment that satisfies the constraints. The set of available constraints provided by a solver is called the modeling language. As users are often interested in a specific solution which is described by an optimization condition, a modeling language called constraint programming (CP) is developed to solve CSPs [108]. Some applications are as follows:

  - Compiler - scheduling, combinational equivalence checking, etc.
  - Supply chain - planning

- **Boolean satisfiability (SAT).** Consider a Boolean (propositional) logic expression consisting of Boolean variables, parentheses, and the operators AND (conjunction), OR (disjunction), and NOT (negation). A literal is a Boolean variable or its negation. A clause is a disjunction of literals. A Boolean expression is a finite conjunction of clauses. The SAT problem consists of finding a Boolean assignment to all variables such that the given expression is true, or that no such assignment exists. Some applications are as follows:
  - EDA - model checking, automatic test pattern generation, etc.
  - Compiler - scheduling, combinational equivalence checking, etc.
  - Supply chain - planning
- **Expression simplification.** The problem of simplification has two aspects: 1) obtaining equivalent but simpler objects; 2) computing unique representations for equivalent objects. Let T be a class of linguistic objects, and S be an effective procedure. The first problem can be formulated as finding a function S that maps T to T and meets $S(t) \sim t, S(t) \leq t$ for all objects t in T. The second problem consists in finding an S that maps T to T and meets $S(t) \sim t, s \sim t \Rightarrow S(s) = S(t)$ for all objects s, t in T. Some applications are as follows:
  - ML compiler - computing expression simplification
  - EDA - functional verification

2.2 Three Paradigms of Algorithmic Design for Combinatorial Optimization

In this section, we introduce three paradigms of algorithmic design for combinatorial optimization, including theoretical, heuristic, and ML algorithms. As an illustrating example, we consider the TSP [40] that is known to be NP-complete, which means that until now no exact polynomial-time algorithm has been designed yet and if such exact approach were ever to be proposed, then it will result in $P = NP$.

As TSP has a lot of real-world applications in planning, logistics, manufacture of microchips, etc, there are lots of theoretical studies that focus on this problem. One main theoretical approach is to design **exact algorithms** to deal with this problem. The most direct solution is to try all permutations of vertices to check which one has the minimum weight. The running time for this approach lies within a polynomial factor of $O(n!)$, which means that the algorithm will take a long time even if there are only 20 vertices. The Held–Karp algorithm [50], which is a dynamic programming-based algorithm, solves the problem in time $O(n^2 2^n)$. Other algorithms, such as various branch-and-bound algorithms [75, 101], solve this problem with a better running time performance. But it is still not known if an exact algorithm for TSP that runs in time $O(1.9999^n)$ exists [116].

Another main theoretical approach is to design **approximation algorithms** that run in polynomial time, while approximating the optimal solution. In general, if there are no restrictions on the edge weights, TSP cannot be approximated within any factor of $poly(n)$, unless $P = NP$ [100]. But in practice, lots of TSP applications are on metric space, that is to say, the edge weights satisfy the triangle inequality. In this case, the algorithm of Christofides [25] approximates it within 1.5, which is known as the best upper bound for metric TSP. The state-of-the-art lower bound for metric TSP is 123/122 [65]. For Euclidean TSP, there is a polynomial-time approximation scheme (PTAS) [5, 89].

Exact algorithms that output an optimal solution have an advantage in performance, however, at the cost of time efficiency. Approximation algorithms have an advantage in the running time, but their worst-case or even practical performance might be very poor. In some cases, implementing existing approximation algorithms that have the best approximation guarantee is difficult. It is interesting to investigate the key ideas of such algorithms for practical implementation.

It is sometimes the case to approach and solve the problem **heuristically**, especially when the problem instance is of a large scale. A heuristic algorithm finds a sub-optimal solution, and delivers a good performance in practice, although without a theoretical guarantee. There is no

general framework for designing a good heuristic algorithm that works for any problem and finds a solution with good performance. Nevertheless, various heuristic search techniques have been developed and helped us obtain good solutions to difficult combinatorial optimization problems. Among them is a promising technique called metaheuristics, which tries to combine basic heuristic methods in higher-level frameworks in order to efficiently explore the set of feasible solutions to a given combinatorial problem. Some examples include simulated annealing [69], tabu search [42–44], evolutionary algorithms like genetic algorithms [46], ant colony optimization [33, 32], iterated local search [82], variable neighborhood search [92], scatter search [45], path-relinking [45], and GRASP (Greedy Randomized Adaptive Search Procedures) [39].

Different from traditional theoretical or heuristic approaches that are human-designed and have a provably guarantee, **ML algorithms** usually learn a model automatically and may have a better performance but require a large training dataset. In recent years, ML scientists have been investigating how to incorporate ML to combinatorial optimization; see surveys [114, 87, 19] for a summarization. For instance, in the TSP problem, there have been several investigations on incorporating different ML models, e.g., pointer network [81, 84], attention [96], transformer [71, 30, 117], structure2vec [68], and GNN [102, 99]. However, current ML algorithms still have several limitations. For example, most of the models have scalability issues due to the complexity of combinatorial optimization problems. Also, most of the existing works involve training an ML model to output solutions directly from the input instance. There have not been many investigations on combining ML algorithms with the existing theoretical/heuristic approaches mentioned earlier.

2.3 Combination of Paradigms for Combinatorial Optimization

We then introduce existing combinations of different paradigms. We first present several famous combinations of theoretical and heuristic algorithms, as they have been the trend before the rise of ML. Then in the following sections, we focus on how traditional theoretical/heuristic algorithms are combined with ML algorithms.

Traditional theoretical analysis always looks at the performance of an algorithm from a worst-case point of view. But in practice, these worst-case scenarios may not occur. As an example, we take a look at the linear programming problem, in which the goal is to maximize a linear function subject to linear constraints. There are some algorithms developed to solve this problem: the simplex method [28], the ellipsoid method [67], the interior point method [64], etc. We know from theory that the simplex method may run in exponential time in the worst case [70, 94], whereas the interior point method and the ellipsoid method run in polynomial time [67, 64]. But in fact, suitably optimized versions of the simplex method remain the most commonly used algorithms in practice. The empirical performance of the simplex method is excellent, most commonly with the running time being linear to the number of variables. The most satisfying explanation comes from the "smoothed analysis theory" developed by Spielman and Teng [113]. The takeaway from this theory is that the simplex method provably runs in polynomial time on "almost all" instances.

3 Combination of Theoretical Algorithms and ML

Traditional algorithms always have little information about the environment. Although there are studies that focus on the environment having good properties, the assumptions are usually so strong that they are not consistent with the actual situation. On the other hand, artificial intelligence (or specifically machine learning), which is currently developing at a tremendous speed, learns and predicts the world well. In general, one can hope that ML, which knows the world well, can help us design algorithms with better performance. This brings us to a hot topic in recent years, called "algorithms with predictions" [91], with applications on counting sketches [53], scheduling [103], online matching [31], etc. The high-level idea is that the algorithm uses an ML algorithm that provides us with predictions about the input, and uses the prediction to make a more effective algorithm; see Figure 2 for the structure. For theoretical analysis, one should analyze the performance of the algorithm as a function of how accurate the prediction is; ideally, the better the prediction, the better the performance. In Section 3.1, we survey recent works on algorithms with predictions in detail.

Specific for solving combinatorial optimization problems, there is another similar research direction called "learning to optimize", which looks deeper into how the learning procedure itself influences the algorithm designs. Intuitively, before solving combinatorial optimization problems, one

Figure 2 ML model augments a theoretical/heuristic algorithm with valuable pieces of information

usually builds a model or fixes an objective function f, and then designs algorithms on model inputs. Here, the inputs vary among different models, for instance, a given instance with all model parameters, e.g., a weighted graph for the shortest path/TSP/minimum spanning tree problems, or an implicit value oracle of f, e.g., a coverage number oracle for the coverage problem. However, in practice, we may not have enough information to obtain accurate inputs, for instance, we may only learn/estimate the passing time of traffic roads by historical data which contains randomness or noises. Consequently, it is interesting to design "robust" optimization algorithms on learned instances that contain noises. In Section 3.2, we survey recent works on learning to optimize in detail.

Another idea of using ML to configure theoretical algorithms is to guide the selection of hyperparameters [20, 51]. Complex optimization algorithms usually have a set of hyperparameters to be selected during optimization, e.g., the learning rate/step size of (stochastic) gradient descent. Carefully selecting these parameters can dramatically improve the performance of the optimization algorithms, and hence, we may use ML to learn good parameters for given instances. As there are few theoretical analyses on this direction, we will not survey them in detail.

3.1 Algorithms with Prediction

3.1.1 Online Algorithms with Prediction

Online algorithms, whose quality is usually measured by the competitive ratio, deal with uncertainty from future input data. The competitive ratio of an algorithm is defined as the worst-case ratio of the algorithm cost to the offline optimum. In our setting, this is a function $c(\eta)$ of the error η of the predictor. We say that an algorithm is γ-robust if $c(\eta) \leq \gamma$ for all η, and that it is β-consistent if $c(0) = \beta$.

- **Consistency.** The performance of the algorithm should improve with better predictions, i.e., the algorithm should actually use the predictor.
- **Robustness.** The performance of the algorithm should degrade gracefully with bad predictions. More specifically, the algorithm performance should be bounded even with a bad predictor.

Ski rental. The first online problem equipped with predictions is ski rental [104], in which a skier is going to ski for an unknown number of days and on each day can either rent skis at unit price or buy them for a higher price b so that there is no need to rent from then on. The uncertainty is in the number of skiing days, which a predictor can estimate. The best known deterministic algorithm for ski rental is the break-even algorithm: rent for the first $b-1$ days and buy on day b. It is easy to observe that the break-even algorithm has a competitive ratio of 2 and no deterministic algorithm can do better. On the other hand, [63] designed a randomized algorithm that yields a competitive ratio of $\frac{e}{e-1} \approx 1.58$, which is also optimal.

Let x be the actual number of skiing days, which is unknown to the algorithm, and y be the predicted number of days. Then $\eta = |y - x|$ is the prediction error. Let $\lambda \in (0, 1)$ be a hyperparameter. For the ski rental problem with a predictor, [104] first obtained a deterministic online algorithm that is $(1 + 1/\lambda)$-robust and $(1 + \lambda)$-consistent. The idea of the algorithm is to trust the prediction cautiously, that is if $y \geq b$, then buy on day $\lceil \lambda b \rceil$, otherwise buy on day $\lceil b/\lambda \rceil$. They next improved these bounds by obtaining a randomized algorithm that is $\left(\frac{1}{1-e^{-(\lambda-1/b)}}\right)$-robust and $\left(\frac{\lambda}{1-e^{-\lambda}}\right)$-consistent, where b is the cost of buying.

(Non-clairvoyant) Job scheduling. The second problem considered is non-clairvoyant job scheduling. In this problem a set of jobs, all of which are available immediately, have to be scheduled on one machine; any job can be preempted and resumed later. The objective is to minimize the sum of completion times of the jobs. The uncertainty in this problem is that the scheduler does not know the running time of a job until it actually finishes. Note that a predictor in this case can predict the running time of a job, once again, by building a model based on the characteristics of the job, resource requirements, and its past behavior. Non-clairvoyant job scheduling, introduced by Motwani et al. [93], is a basic problem in online algorithms with a rich history and, in

addition to its obvious applications to real-world systems, many variants and extensions of it have been studied extensively in the literature [16, 15, 55, 54]. Motwani et al. [93] showed that the round-robin algorithm has a competitive ratio of 2, which is optimal. [104] obtained a randomized algorithm that is $(2/(1-\lambda))$-robust and $(1/\lambda)$-consistent. [56] proposed another error measure because the ℓ_1 error measure violates a natural and desirable Lipschitz-like property for the total completion time objective. The new error measure better captures the sensitive nature of the objective and allows the researchers to obtain an algorithm with a competitive ratio of at most $(1+\epsilon)opt + O_\epsilon(1) \cdot v(p, \hat{p})$, where $v(\cdot,\cdot)$ is the new error measure.

Online caching (paging). The online caching (or paging) problem considers a system with two levels of memory: a fast memory of size k and a slow memory of (almost) infinite size. A caching algorithm is faced with a sequence of requests for elements. If the requested element is in the fast memory, a cache hit occurs and the algorithm can satisfy the request at no cost. If the requested element is not in the fast memory, a cache miss occurs, the algorithm fetches the element from the slow memory, and places it in the fast memory before satisfying the request. If the fast memory is full, then one of the elements must be evicted. The eviction strategy forms the core of the problem. The goal is to find an eviction policy that results in the fewest number of cache misses.

Several $O(k)$-competitive deterministic and $O(\log k)$-competitive randomized algorithms are known [1] , as well as lower bounds of $\Omega(k)$ and $\Omega(\log k)$ on the competitive ratio.

Online caching problems have been extended to allow an oracle that imperfectly predicts the next arrival of a requested element. [83] demonstrated that following the oracle's recommendations may lead to very poor performance, even when the average error is quite low. They also showed how to modify the Marker algorithm to take into account the oracle's predictions, and proved that this combined approach achieves a competitive ratio of $O(1+\min(\sqrt{\eta/opt}, \log k))$ when the overall ℓ_1 prediction error is bounded by η, and opt is the cost of the optimal offline algorithm. This ratio decreases as the oracle's error decreases, and is always capped by $O(\log k)$, which can be achieved without any oracle input. [106] provided an improved algorithm with a competitive ratio of $O(1+\min((\eta/opt)/k, 1)\log k)$ and a lower bound of $\Omega(\log \min((\eta/opt)/(k \log k), k))$. Recently, [115] obtained a deterministic algorithm with a competitive ratio of $2\min(\min(1+2\eta/opt, 2+4\eta/((k-1)opt)), k)$, and a randomized algorithm with a competitive ratio of $(1+\epsilon)\min(\min(1+2\eta/opt, 2+4\eta/((k-1)opt)), H_k)$ where H_k is the k-th harmonic number and the trade-off in ϵ and the additional cost is additive. [115] also showed that the competitive ratio bound for any deterministic learning-augmented online caching algorithm must be at least $1+\Omega(\min(\eta/(k \cdot opt), k))$.

Value-maximization secretary problem. In the secretary problem there is a set $\{1, \cdots, n\}$ of secretaries, each with a value $v_i \geq 0$ for $i \in 1, ..., n$, that arrive in a uniformly random order. Whenever a secretary arrives, we have to irrevocably decide whether we want to hire that person. If we decide to hire a secretary, we automatically reject all subsequent candidates. The goal is to select the secretary with the highest value. There are two versions of the secretary problem. In the classical secretary problem, the goal is to maximize the probability with which the best secretary is chosen. There is a slightly different version, where the goal is to maximize the expected value of the chosen secretary, which is the so-called value-maximization secretary problem. The (optimal) solution [77, 37] to both variants of the secretary problem is to first observe a fraction of n/e secretaries. After that, the first secretary with a value higher than the best value seen in the first fraction is selected. This yields a $1/e$-approximation for both versions.

The machine-learned advice in this problem is a prediction $p*$ for the maximum value $OPT = \max_i v_i$ among all secretaries, and not which secretary has the highest value. [4] showed that for any $\lambda \geq 0$ and $c > 1$, there is a deterministic algorithm for the (value-maximization) secretary problem that is asymptotically $g_{c,\lambda}(\eta)$-competitive in expectation, where

$$g_{c,\lambda}(\eta) = \begin{cases} \max\{\frac{1}{ce}, [f(c)(\max\{1-\frac{\lambda+\eta}{opt}, 0\})]\} & \text{if } 0 \leq \eta < \lambda \\ g_{c,\lambda}(\eta) = \frac{1}{ce} & \text{if } \eta \geq \lambda \end{cases}.$$

The function $f(c)$ is given in terms of the two branches W_0 and W_{-1} of the Lambert W-function and reads $f(c) = \exp\{W_0(-1/(ce))\} - \exp\{W_{-1}(-1/(ce))\}$.

Online bipartite matching. The online bipartite matching problem is the problem in which the set of nodes L of a bipartite graph $G = (L \cup R, E)$, with $|L| = n$ and $|R| = m$, arrives online in a uniformly random order [73, 66]. Upon arrival, a node reveals the edge weights to its neighbors in R. We have to irrevocably decide if we want to match up the arrived online node with one of its (currently unmatched) neighbors in R. Kesselheim et al. [66] gave a tight $1/e$-competitive deterministic algorithm for this setting that significantly generalizes the same guarantee for the classical secretary algorithm [77, 37]. The prediction in this setting is a vector of values $p* = (p*_1, \ldots, p*_m)$ that predicts the edge weights adjacent to the nodes $r \in R$ in some fixed

optimal (offline) bipartite matching. That is, the prediction $p*$ indicates the existence of a fixed optimal bipartite matching in which each node $r \in R$ is adjacent to an edge with weight $p*_r$. The prediction error is then the maximum prediction error taken over all nodes in $r \in R$ and minimized over all optimal matchings. This generalizes the prediction used for the classical secretary problem. This type of prediction closely corresponds to the vertex-weighted online bipartite matching problem [2]. [4] showed that for any $\lambda \geq 0$ and $c > d \geq 1$, there exists a deterministic algorithm for the online bipartite matching problem that is asymptotically $g_{c,d,\lambda}(\eta)$-competitive in expectation, where

$$g_{c,d,\lambda}(\eta) = \begin{cases} \max\{\frac{1}{c}\ln(\frac{c}{d}), [\frac{d-1}{2c}(\max\{1 - \frac{(\lambda+\eta)|\psi|}{opt}, 0\})]\} & \text{if } 0 \leq \eta < \lambda \\ \frac{1}{c}\ln(\frac{c}{d}) & \text{if } \eta \geq \lambda \end{cases}$$

and $|\psi|$ is the cardinality of an optimal (offline) matching ψ of the instance.

Bin packing. Bin packing is a classic optimization problem and one of the original NP-hard problems. In the online variant, the set of items is not known in advance, but is rather revealed in the form of a sequence. Upon the arrival of a new item, the online algorithm must either place it into one of the currently open bins, as long as this action does not violate the bin's capacity, or place it into a new bin. Assume that the size of each item is an integer in $[1, k]$, where k is the bin capacity. This is a natural assumption on which many efficient algorithms for bin packing rely, e.g., [27, 41, 110]. Furthermore, without any restriction on the item sizes, [88] showed that no online algorithm with advice of size sublinear in the size of the input can have competitive ratio better than 1.17 (even if the advice is error-free). This negative result implies that some restriction on item sizes is required so as to leverage frequency-based predictions. Consider an input sequence σ. For any $x \in [1, k]$, let $n_{x,\sigma}$ denote the number of items of size x in σ. Define the frequency of size x in σ, denoted by $f_{x,\sigma}$, to be equal to $n_{x,\sigma}/n$, hence $f_{x,\sigma} \in [0, 1]$. Algorithms use these frequencies as predictions. Namely, for every $x \in [1, k]$, there is a predicted value of the frequency of size x in σ, which is denoted by $f'_{x,\sigma}$. The predictions come with an error, so in general, $f'_{x,\sigma} \neq f_{x,\sigma}$. To quantify the prediction error, let f_σ and f'_σ denote the frequencies and their predictions in σ, respectively, as points in the k-dimensional space. In line with previous work on online algorithms with predictions, e.g., [104], the error η is the L1 norm of the distance between f_σ and f'_σ.

Metrical task systems. Initially, we are given a metric space M of states, which can be interpreted, for example, as actions, investment strategies, or configurations of some production machine. We start at a predefined initial state x_0. At each time $t = 1, 2, \ldots$, we are presented with a cost function $\ell_t : M \mapsto R^+ \cup \{0, +\infty\}$ and our task is to decide either to stay at x_{t-1} and pay the cost $\ell_t(x_{t-1})$, or to move to some other (possibly cheaper) state x_t and pay $dist(x_{t-1}, x_t) + \ell_t(x_t)$, where $dist(x_{t-1}, x_t)$ is the cost of the transition between states x_{t-1} and x_t. The objective is to minimize the overall cost incurred over time. At each time t, the predictor produces a prediction p_t of the state where the algorithm should be at time t. The prediction error with respect to some offline algorithm OFF is

$$\eta = \sum_{t=1}^{T} \eta_t;\ \eta_t = dist(p_t, o_t),$$

where o_t denotes the state of OFF at time t and T denotes the length of the input sequence. [3] proved two general results. First, let A be a deterministic α-competitive online algorithm for a problem P belonging to a metrical task system (MTS). There is a prediction-based deterministic algorithm for P achieving a competitive ratio of $9 \cdot \min\{\alpha, 1 + 4\eta/OFF\}$ against any offline algorithm OFF, where η is the prediction error with respect to OFF. Second, assume that A is a randomized α-competitive online algorithm for an MTS P with metric space diameter D. For any $\epsilon \leq 1/4$, there is a prediction-based randomized algorithm for P achieving a cost below $(1 + \epsilon) \cdot \min\{\alpha, 1 + 4\eta/\text{OFF}\} \cdot \text{OFF} + O(D/\epsilon)$, where η is the prediction error with respect to OFF. Thus, if OFF is (near-) optimal and $\eta \leq$ OFF, the competitive ratio is close to $1 + \epsilon$.

3.1.2 Stream Algorithms with Prediction

The data stream model is a fundamental model for processing massive data sets with limited memory and fast processing time. [60] explored the full power of an oracle, called heavy hitter oracle, which predicts whether a data element will appear more frequently than some given threshold. They show that the oracle can be applied to a wide array of problems in data streams, sometimes resulting in the first optimal bounds for such problems, and sometimes bypassing known lower bounds without such an oracle. They applied the oracle to count distinct elements on the difference of streams, estimate frequency moments, estimate cascaded aggregates, and estimate moments of geometric data streams. For the distinct elements problem, this technique bypasses the known lower bound and obtains a new space-optimal algorithm. For estimating the p-th frequency moment for $0 < p < 2$, this work obtains the first

algorithms with optimal space and update time. For estimating the p-th frequency moment for $p > 2$, they obtained a quadratic savings in memory, bypassing known lower bounds without an oracle.

In the data stream model, we assume there is an underlying frequency vector $x \in \mathbb{Z}^n$ initialized to 0^n, which evolves throughout the course of a stream. The stream consists of updates of the form (i, w), meaning $x_i \leftarrow x_i + w$. Suppose M is such that $||x||_\infty = \max_{i\in\{1,\ldots,n\}} |x_i| \leq M$ throughout the stream. At the end of the stream, we are asked to approximate $f(x)$ for a function $f : \mathbb{Z}^n \to R$. Because of the sheer size of the stream, most algorithms are approximate and randomized. Typically, such algorithms output an estimate Z for which $P\{(1-\epsilon)f(x) \leq Z \leq (1+\epsilon)f(x)\} \geq 2/3$, where the probability is over the randomness used by the algorithm, and not of the input stream, which may be a worst-case stream. The success probability $2/3$ can be replaced with any constant greater than $1/2$ by independently repeating the algorithm a constant number of times and taking the median estimate. The space complexity of the algorithm is measured in bits and the goal is to use much less than the trivial n bits of space required to store x.

We consider estimating the following common functions $f(x)$ in the data stream model.

- Distinct Elements: $f(x) = ||x||_0$, where $||x||_0$ is the number of nonzero coordinates of x, defined as $||x||_0 = |\{i : x_i \neq 0\}|$. This quantity is useful for detecting denial-of-service attacks and for database query optimization [61].
- F_p-Moment: $f(x) = ||x||_p^p$, where the ℓ_p-norm of x is defined as $||x||_p = (\sum_i |x_i|^p)^{1/p}$. For $0 < p < 2$, these are often more robust than the Euclidean norm [57]. For $p > 2$, these are used to estimate skewness and kurtosis [58].
- (k, p)-Cascaded Norm: In this problem x is an $n \times d$ matrix that receives updates to its individual entries, and $f(x) = ||x||_{k,p} := (\sum_i (\sum_j |x_{ij}|^p)^{k/p})^1/k$. These are useful for executing multiple database queries [26, 59].

We assume access to a heavy hitter oracle, which receives an input $i \in [n]$, and outputs whether x_i will be a heavy hitter at the end of the stream. The definition of heavy hitter varies for different data stream problems. There are two kinds of oracles. The first kind indicates whether or not $|x_i| \geq T$ and the second kind indicates whether or not $|x_i|^p \geq \frac{1}{T}||x||_p^p$, where T is a pre-determined threshold associated with the problem. We shall use an oracle of the first kind for the distinct elements problem and oracles of the second kind for all other problems.

The current best algorithm for estimating $L_0 = ||x||_0$, in the presence of additions and deletions to coordinates of x, is due to [61]. [61] showed that with the trained oracle, there is an algorithm for $(1 \pm \varepsilon)$-approximating L_0 using space $O(\varepsilon^{-2}(\log n)log(1/\varepsilon))$ with a success probability of at least $2/3$, and with $O(1)$ update and reporting times.

[61] also showed the following results.

Under the perfect assumption of a heavy hitter oracle, one can estimate $||x||_p^p$ within a factor $1 \pm 2\varepsilon$ in $O(\varepsilon^{-4}n^{1/2-1/p}\log(n)\log(M))$ bits with a success probability of at least $3/5$. If the probability that the oracle gives us an incorrect prediction is $\delta > 0$, then we can estimate $||x||_p^p$ within a factor $1 \pm 2\varepsilon$ in $O(\varepsilon^{-4}(n)^{1/2-1/p}\log(n)\log(M))$ bits of space when $\delta = O(1/\sqrt{n})$, or in $O(\varepsilon^{-4}(n\delta)^{1-2/p}\log(n)\log(M))$ bits of space otherwise. Both guarantees hold with a success probability of at least $3/5$.

Let $0 < p < 2$ and $0 < \varepsilon < 1/2$. With a perfect heavy hitter oracle, there exists a randomized algorithm which outputs $(1 \pm 3\varepsilon)||x||_p^p$ with a probability of at least 0.7 using $O(\varepsilon^{-2}\max\log(nmM), \log^3(1/\varepsilon)\log\log(1/\varepsilon))$ bits of space. The expected amortized update time is $O(1)$. The reporting time is $O(1/\varepsilon^2)$. In the case of a noisy oracle, we assume that for a fixed s, the oracle will not identify more than $O(s)$ items x_i as heavy hitters (meaning that $|xi| > \frac{1}{s}||x||_p^p$). Suppose that the heavy hitter oracle errs with probability $\delta = O\left(\frac{\varepsilon^2}{\log^2(1/\varepsilon)\log\log(1/\varepsilon))}\right)$. There exists an algorithm with the same guarantee.

For cascaded norms, let $\varepsilon > 0$ and $k \geq p \geq 2$ be constants. There exists a randomized algorithm which receives an $n \times d$ matrix x in a stream of additions and deletions to its coordinates, and outputs a $(1+\varepsilon)$-approximation to $||x||_{k,p}$, with a probability of at least $2/3$ using $\tilde{O}(n^{1-\frac{1}{k}-\frac{p}{2k}}d^{\frac{1}{2}-\frac{1}{p}})$ space.

Besides, under the assumption of a heavy hitter oracle, there is a rectangle-efficient singlepass streaming algorithm which outputs a $(1 \pm \varepsilon)$-approximation to $||x||_p^p$ with the probability $1 - \delta$ for $p > 2$. It uses $O^*(\Delta^{d(1/2-1/p)})$ bits of space and $O^*(\Delta^{d(1/2-1/p)})$ time to process each rectangle in the stream.

3.2 Learning to Optimize

In this section, we review the current progress on the topic of "learning to optimize" through two main directions, "optimization from samples" and "optimization with noises". In the following, we mainly introduce the mathematical

models of these directions, as well as the impossibility results and algorithmic results under different settings.

3.2.1 Optimization from Samples

In this section, we regard historical data as samples drawn from a certain underlying distribution, and our goal is to study how to optimize objective functions from sample data. We consider (constrained) combinatorial optimization problems whose form is

$$\max_{S\in\mathcal{M}} f(S),$$

where $f: 2^N \to \mathbb{R}$ is an objective function and $\mathcal{M} \subseteq 2^N$ is a constrained collection. And suppose there exists a value oracle O_f that returns value $f(S)$ for a subset $S \subseteq [N]$. To evaluate the ability of samples for optimization, Balkanski et al. [11] first propose the following model called *optimization from samples (OPS)*.

Definition 3.1 (OPS model). *Given a function $f: 2^N \to \mathbb{R}$, we say $\mathcal{S} := \{S_i, f(S_i)\}_{i=1}^t$ is a collection of samples, where each S_i is i.i.d. drawn from a certain underlying distribution $\mathcal{D}$. Given parameter $\alpha \in (0,1]$, we say a function family $\mathcal{F}: 2^N \to \mathbb{R}$ is α-approximable with respect to distribution $\mathcal{D}$ and constraint $\mathcal{M}$ if there exists an algorithm A that given error parameter $\delta \in (0,1)$ and a collection $\mathcal{S}$ of samples as inputs, outputs $S \in \mathcal{M}$ satisfying that*

$$\Pr_{S_1,\dots S_t\sim\mathcal{D}^t}\left[\mathbb{E}_A[f(S)] \geq \alpha \cdot \max_{T\in\mathcal{M}} f(T)\right] \geq 1-\delta.$$

Here, α is called the approximation ratio and t is the sample complexity. Also note that we do not require A to be a polynomial algorithm, because we concentrate on the power of samples under the OPS model.

Obviously, the distribution $\mathcal{D}$ heavily affects the approximability of $\mathcal{F}$. For instance, if $\mathcal{D}$ always returns an empty set or a fixed subset, then we cannot expect to learn enough information for optimization from the samples. Hence, one would like to consider "reasonable" distributions, and investigate whether approximable functions under the query model O_f are still approximable under the OPS model, e.g., submodular function. To this end, people consider a function family called *probably mostly approximately correct learnability (PMAC)*, proposed by Balcan et al. [8]. Roughly speaking, we say a function f is PMAC-learnable with respect to some distribution $\mathcal{D}$, if for an arbitrary subset $S \sim \mathcal{D}$, we can learn $f(S)$ with high probability by a collection $\mathcal{S}$ of samples. PMAC contains a large amount of function families, including coverage functions and influence functions, which are the most studied functions under the OPS model. We first define the coverage function.

Definition 3.2 (Coverage maximization). *Given a bipartite graph $G = (L, R, E)$ where L and R represent the left and right set of nodes, and E represents the edge set between L and R. A coverage function $f: 2^L \to \mathbb{R}_{\geq 0}$ is defined to be the number of neighbors of a subset $S \subseteq L$, i.e., $f(S) = |N(S)|$. The coverage maximization problem is to select a subset $S \subseteq L$ of size $k \geq 1$ that maximizes $f(S)$, i.e., $\max_{S\subseteq L:|S|=k} f(S)$.*

Note that the coverage maximum problem is PMAC-learnable [6] and admits a $(1-1/e)$-approximation greedy algorithm under the query model [98]. Next, we define the influence function, which can be regarded as a generalization of the coverage function to general directed graphs and is widely used in social networks.

Definition 3.3 (Influence maximization). *Given a directed graph $G = (V, E, p)$ where V is the node set, E is the edge set, and p_e is the probability vector on each edge $e \in E$, the state of each node is either active or inactive. Given a seed set S_0 as active at time $t = 0$, we activate other nodes as follows: At time $t = 1, 2, \dots$, first let $S_t = S_{t-1}$, and then for any $v \notin S_{t-1}$, let $N^{in}(v)$ denote the collection of nodes u with $(u, v) \in E$. Each node $u \in N^{in}(v) \cap (S_{t-1} \setminus S_{t-2})$ activates v with probability p_{uv} independently, and finally all active nodes v are added to S_t. Consequently, we have a sequence of active subsets $S_0 \subseteq S_1 \subseteq S_2 \subseteq \cdots$ until there are no new active nodes. Given S_0, define $\phi(S_0)$ to be the collection of final active nodes and define influence function $f: 2^V \to \mathbb{R}$ to be $f(S) = \mathbb{E}[|\phi(S)|]$, that is, the expected active number of seed set S_0. The influence maximization problem is to select a seed set S of size at most k such that $f(S)$ is maximized.*

Note that both the coverage function and the influence function are submodular.[1]

Intuitively, we may divide the OPS model into two steps:

1. Construct a function $\tilde{f}$ as an estimation of f from samples by the PMAC-learnable guarantee.
2. Optimize the original problem under the query model of $\tilde{f}$.

[1] We say f is submodular if for any $S \subseteq T \subseteq N$ and $u \notin T$, we have $f(S \cup u) - f(S) \geq f(T \cup u) - f(T)$.

However, we will see that although both two steps are not hard to implement, their combination may not provide a reasonable (approximation) algorithm. Next, we introduce the current impossibility and algorithmic results under the OPS model, specifically including the coverage maximization problems and the influence maximization problems.

Impossibility results. Balkanski et al. [11] first studied the coverage maximum problem under the OPS model and presented the following theorem.

Theorem 3.4 (Impossible result for coverage maximization [11]). *For any distribution $\mathcal{D}$, there is no algorithm for coverage maximization whose approximation is better than $n^{-1/4}$ that observes fewer than exponentially many samples and this bound is tight.*

This result is somehow surprising since coverage maximization is proven to be PMAC-learnable [6]. The main idea is to construct a class of functions that can be learned well on most subsets (with respect to $\mathcal{D}$), but cannot be learned well on (approximate) optimal solutions, and consequently, samples do not provide enough information for optimization.

Impossibility results also appear in other problem settings. Balkanski et al. [12] constructed a class of submodular minimization problems $f : 2^N \to [0, 1]$ and proved that no algorithm with polynomial samples can provide an additive $(1/2 - O(1))$-approximation. This result can be extended to convex minimization [13]. Overall, we note that the approximability under the query model and the OPS model can be very different, even if the function is PMAC-learnable. In the following, we show the current progress that aims to overcome the impossibility results provided above.

Algorithmic results. In general, there are three types of attempts. The first attempt is to assume f satisfies additional properties. For instance, Balkanski et al.[10] considered monotone submodular functions with curvature $c \in [0, 1]$.[1] They provided a tight $(1 - c)/(1 + c + c^2)$-approximation algorithm. This matches the intuition where a linear function can be solved under the OPS model. However, the curvatures for both the coverage and the influence functions are 1. For the influence maximization problem, Balkanski et al. [9] assumed that the social network G is generated by the stochastic block model and presented a constant approximation algorithm. Overall, this attempt does not change the OPS model and hence, we need to design specific properties for different optimization problems.

The second attempt is to weaken the objective of the OPS model. Instead of optimizing over the whole function, Rosenfeld et al. [107] proposed a DOPS (distributional optimization from samples) model, whose goal is to use a small collection of samples to find the optimized sample among another unknown large collection of samples drawn from the same distribution. They proved the equivalence between approximable under the DOPS model and PMAC-learnable. This attempt is to provide an explanation of PMAC-learnable under the sampling model. However, one may still want to achieve the global optimal with respect to reasonable distributions $\mathcal{D}$.

The third attempt is to assume that we not only have the values $f(S_i)$ of samples S_i, but also obtain additional structured information. Chen et al. [23] first investigated this approach on coverage maximization, and proposed the following OPSS (optimization from structured samples) model: Each sample is represented by $(S_i, N_G(S_i))$ where $N_G(S_i)$ is the collection of neighbors of S_i. They proved that the coverage maximum problem admits an $O(1)$-approximation under the OPSS model if $\mathcal{D}$ satisfies some mild assumption (e.g., $\mathcal{D}$ is some uniform distribution). Further, Chen et al. [24] extended this result to the influence maximization problem, in which each sample is of the form $(S_{i,0}, S_{i,1}, \ldots, S_{i,n-1})$ that consists of the full passing path. Assuming that $\mathcal{D}$ is a product distribution, they provided a constant approximation algorithm. In conclusion, we observe that the additional structured information may complete the gap between learnable and approximable. Moreover, the structured information is varied as the problem changes, and it is interesting to investigate more suitable structured information for other problems.

3.2.2 Optimization with Noises

As discussed above, the main difficulty under the OPS model seems to be that we may not be able to estimate the value of the optimal solution from samples. Then one might wonder whether optimization can be done if we can estimate all subsets. This opens another direction called optimization with noises, in which we can learn a function $\tilde{f}$ as an estimation of underlying objective f such that for any $S \subseteq 2^N$, $\tilde{f}(S) \in (1 \pm \varepsilon) f(S)$. Here, $\varepsilon \in (0, 1)$ is a noisy parameter and we usually want a small ε. Such an

[1] Curvature measures how much a submodular function is close to linear. As curvature is close to 0, the submodular function is more close to linear.

estimation $\tilde{f}$ can be learned from historical samples, e.g., through a neural network or constructing a sketch. Surprisingly, even if we have such $\tilde{f}$, the optimization problem may still be hard to solve. In the following, we discuss two settings: erroneous oracle and noisy oracle.

Erroneous oracle. Hassidim et al. [48] proposed the following erroneous oracle in which for any $S \in 2^N$,

$$\tilde{f}(S) = \varepsilon_S \cdot f(S),$$

where $\varepsilon_S \in [1-\varepsilon, 1+\varepsilon]$. Here, ε_S can be selected by the adversary. They proved the following theorem that shows that submodular maximization is hard with an erroneous oracle.

Theorem 3.5 (Impossibility results with erroneous oracles [48]). *No randomized algorithm can obtain an $n^{-0.5+\delta}$-approximation for maximizing a monotone submodular function under a cardinality constraint with less than an exponential number of queries to an ε-erroneous oracle.*

Their main idea is to construct two functions f_1 and f_2 that are close in almost all subsets $S \in 2^N$, say $f_1(S) \in (1 \pm \varepsilon) f_2(S)$, and hence, the adversary can select noise parameters ε_S such that one cannot distinguish f_1 and f_2 on these subsets. However, the values of the optimal solutions of f_1 and f_2 differ by a lot. Hassidim et al. [48] proved that a polynomial number of queries is not enough to distinguish f_1 and f_2 in this setting.

To get rid of this impossibility result, Hassidim et al. [48] proposed a noisy oracle, in which $\varepsilon_S \sim D$ are i.i.d. drawn instead of being selected by the adversary. They provided a $(1-1/e-\varepsilon)$-approximation using access to a noisy oracle. However, their algorithm requires $n^{O(1/\varepsilon)}$ queries, which still needs to be reduced.

In this section, we review existing progress on optimization from samples and optimization with noises. These work is interesting but there remain several open problems, e.g., more reasonable models are required to get rid of the impossibility results, and how to decrease the sample/query complexity of existing algorithmic results.

4 Combination of Heuristic Algorithms and ML

Next, we focus on the combination of heuristic algorithms and ML. In comparison with applications of ML in theoretical works, one can consider that the heuristic algorithm controls the high-level structure while calling an ML model to assist in lower-level decisions. This learning augmented framework is summarized in Figure 3. Intuitively, it is common that an algorithm is designed to solve a general class of problems, namely being able to take any feasible instance from the class as input and return the desired solution. Furthermore, theoretical studies usually provide analysis for the algorithms' properties, such as convergence rate and approximation ratio, for the worst cases among all problem instances. However, in practice, the performance of an algorithm for each specific instance is usually very sensitive to the choice of the algorithm's parameters and the initial solution (if an initial solution is not preset for the algorithm). Thus, it is natural to consider learning the best instance-dependent control policy for an algorithm, without changing the algorithm itself.

In this section, we name this kind of approach as "learning augmented algorithms", and review current progress on this topic. We shall introduce two representative directions, "learning control policies" and "learning to warm start", and a rising direction called "adaptive learning augmentations".

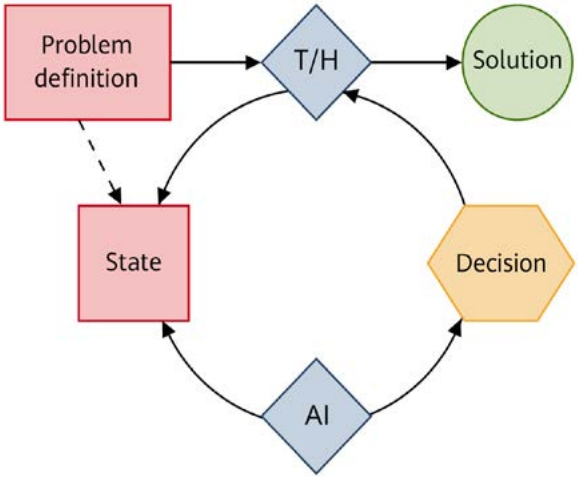


Figure 3 The combinatorial optimization algorithm repeatedly queries the same ML model to make decisions. The ML model takes as input the current state of the algorithm, which may include the problem definition.

4.1 Learning Control Policies

The control policies for an algorithm mainly refer to its hyperparameters, for example, update rate for an iterative algorithm, or searching strategies for a heuristic algorithm. In this section, we introduce an application of the branch-and-bound algorithm as an example, and summarize other similar learning augmented algorithms in the scheme of learning control policies.

Branch-and-bound, BnB for short, is mainly referred to as a paradigm for solving discrete optimization problems. We consider a problem of minimizing a real-valued function f over a feasible set S, where S is discrete. BnB usually starts with a single root state and maintains a queue $\mathcal{L}$ of active nodes, representing subsets of S, as well as global upper/lower bound for the objective. While the queue $\mathcal{L}$ is not empty, BnB pops a node $S_i \in S$ from $\mathcal{L}$ at each step, with some node selection strategy, and computes the bound for the objective restricted to the subset S_i, by applying necessary relaxation and efficient solvers. According to the local and global bounds, BnB distinguishes three possible cases:

1. The local lower bound is larger than the current global upper bound, and as a result it is not possible to obtain a better solution by further branching the subset. Then this subtree (subset) is pruned.
2. The local lower bound and local upper bound match, and then the optimal solution restricted on the subtree (subset) is found.
3. Otherwise, S_i is branched into all its children, which are pushed back onto $\mathcal{L}$.

In cases 2 and 3, the global lower and upper bounds are updated accordingly. The algorithm terminates when $\mathcal{L}$ is empty or the gap between the global lower and upper bounds is small enough.

The most important control policy of a BnB algorithm is the node selection strategy, to be precise, how to choose the next subset from the queue to compute the bounds and decide whether to branch. Generally speaking, one would expect a good strategy to find the optimal solution as quickly as possible. That means the strategy must choose proper branching subsets first, such that good solutions are found early and most unnecessary subtrees (subsets) are pruned. Several strategies are popular in traditional applications, such as the well-known first-in-first-out (FIFO), last-in-first-out (LIFO), and the best-first strategy, which relies on another priority queue, for example sorting the nodes on their local lower bounds.

On the other hand, it is straightforward to consider leveraging some training data for the problem to be solved, and learning a good strategy. Despite the customized representations for the problems as well as the queues, the intuition for the learning procedure is to learn the most possible node in each queue, to which the optimal solution of the problem instance belongs.

Bai et al. [7] applies this paradigm to solve the maximum common subgraph (MCS) between two input graphs. In their design, the subsets to branch are possible node pairs between the input graphs, and the reinforcement learning is applied to learn the node pairs by maximizing the sizes of found common subgraphs. A similar approach is considered by Liu et al. [79] , who set the learning objective as minimizing the branching times. Both of these learning augmented BnBs achieve better performance compared with end-to-end learning based solvers for MCS problems.

He et al. [49] applied this paradigm to solve the mixed integer linear programming (MILP) problem. The subsets to branch are naturally the variables and districts they belong to. In addition to learning the proper branching variable for each state, the authors also learnt a pruning policy, which predicts whether the optimal solution belongs to a subtree (subset). The pruning policy further accelerates the algorithm as the computations of the local upper/lower bounds for most subtrees, which are predicted not to contain the optimal solution, are ignored. With both control policies, BnB achieves better performance in terms of speed and optimal gap, when compared with the SOTA solvers for MILP. Other approaches also design different ML techniques to learn the branching decisions made at every node on the branch-and-bound tree for MILP [68, 80, 52]. Similarly, Kruber et al. [74] use ML on MILP instances to first decide whether or not applying a Dantzig-Wolfe decomposition will be effective. Bonami et al. [21] use ML to decide if linearizing the problem will solve it faster. Meanwhile, the aggressiveness of a MILP solver and the heuristic building framework could also be selected by ML [62, 85]. Further, for quadratic programming (QP), Baltean-Lugojan et al. [14] use ML to select promising cutting planes.

Drawbacks. Applying the learnt control policies, which are usually represented by neural networks, also requires computation resources and time budgets. In the mentioned examples for BnB, the algorithms should call the policies each time they decide a branching node. This means the learnt control policies must improve the performance of the original algorithms a lot, so that the extra cost can be covered.

4.2 Learning to Warm Start

Although many heuristic algorithms construct initial solutions as their first iterative steps, when compared with the control policies, whether the initial solutions are good or not seems to be less important. A similar scenario occurs in theoretical analysis of many algorithms. Except for those

algorithms which specifically identify initial solutions, the properties of convergence and optimality are usually proved for random initialization or arbitrary worse initialization.

However, being able to predict a proper initial solution for a specific problem instance, namely a warm start, definitely helps improve the corresponding algorithm's performance. Intuitively, for example, it will probably take an iterative algorithm less time to obtain the optimal solution if it starts with a solution close enough to the optimal one. This is the exactly the motivation of recent literature. Basically, these works learn prediction models based on training data for some problem instances with their optimal solutions. Then for unseen problem instances, traditional algorithms are applied with warm starts being the predicted solutions.

For example, Bemporad and Naik [18], as well as Masti and Bemporad [86], use this idea to learn warm starts for BnB to solve mixed-integer quadratic programming. Duan et al. [35]. show similar improvements when using a predicted solution as the warm start of traditional algorithms solving Nash equilibrium, in the sense of both computational efficiency and approximation to the optimality.

Drawbacks. The approaches of learning to warm start share the same drawback as "learning control policies", because an extra inference of neural network is needed before solving each problem instance. On the other hand, there are few works aimed at learning the optimal warm starts for a specific algorithm. Theoretically speaking, it is still unknown whether the closer an initial solution is to the optimal solution, the better it is as a warm start.

The opposite direction. An interesting observation is that the idea of "learning to warm start" may also be regarded as augmenting machine learning through traditional algorithms. For example, Li et al. [76] designed a learning framework to solve combinatorial optimization problems, mainly three equivalent problems: maximum clique (MC), maximum independent set (MIS), and minimum vertex cover (MVC). The designed learning framework outputs a group of solutions for each input instance, and the local search algorithm is applied to improve these solutions and finally find the optimal one. However, it is common in traditional solvers of MC/MIS/MVC that use a simple way to generate an initial solution and then apply the local search algorithm. In other words, the learning part of this approach is indeed learning a group of good warm starts for the local search.

4.3 Adaptive Learning Augmentations

The previous two schemes of learning augmentation can be regarded as "static augmentation", because the learning model for the control policies or the warm starts is fixed during the running process of the augmented algorithms. In contrast, a recent work [118] designs a learning procedure that learns the augmentation along with the running process of the algorithm by taking each problem instance as the input. In this section, we briefly introduce the studied problem, augmented heuristic algorithm, and learning framework.

The traveling salesman problem (TSP) is defined on a complete undirected graph, where each edge has a cost representing the distance between its two endpoints. The problem is to find a Hamiltonian route to minimize the total cost of the edges along the route. The Lin-Kernighan-Helsgaun (LKH) algorithm is a famous heuristic based on local search for solving the TSP. Starting from an initial solution, LKH iteratively runs a special local search technique called k-opt. Each k-opt process considers k edges in the current solution, tests a series of other k connections among the $2k$ endpoints of these edges, and replaces the k edges with the optimal choice (minimal total cost). The key in LKH is to choose the k edges according to the α-value (and its extensions). Ignoring the detailed calculation, it is believed that an edge with smaller α-value is more likely to appear in the optimal solution of TSP.

Zheng et al. [118] designed a reinforcement learning procedure, replacing 2-opt process. Specifically, a Q-value of each edge is initiated based on α-value. Then at each step, the state of the system is a node p_i, an action is one of its adjacent edge (p_i, p_{i+1}) chosen according to the Q-value, and the system transits to the next node p_{i+2}, which connects to p_{i+1} in the original solution (this operation is actually to replace two edges (p_{i-1}, p_i) and (p_{i+1}, p_{i+2}) in the previous solution with another two edges (p_i, p_{i+1}) and (p_{i+2}, p_{i-1}) without violating the feasibility). By setting the reward for each state action pair as the improvement of the solution, standard reinforcement learning is applied to update the Q-value.

This approach combines the reinforcement learning technique and traditional heuristic procedure, achieving better performance in solving large TSP instances when compared with LKH. The Q-value plays an important role as an augmentation to α-value. The elegant design of its adaptive update, on the one hand, provides a novel potential for learning augmentations, whereas on the other hand, it relies on a lot of domain information both about the problem structure and the heuristic algorithm itself.

5 Mechanism Design with Learning

Mechanism design can be regarded as a special process of algorithm design, being aware of rational agents. Precisely speaking, we can define an algorithm as designed to be able to take any instance from a specific problem as the input and return the output, satisfying some desired property. The input instances of a designed mechanism are provided by agents, who benefit from the mechanism's outputs. As a result, when designing a mechanism, we have to consider both its objective and the agents' incentive of misreporting.

The natural intuition for mechanism design is to design a game for agents to play, with agents' actions being its input, such that the Nash equilibrium of the game leads to desired output. Fortunately, according to the famous revelation principle, one can focus on designing "truthful mechanisms", under which the equilibrium strategy of each agent is honestly reported. Intuitively, this is because if a mechanism optimizes an objective when agents play equilibrium strategies, then there is an equivalent truthful mechanism taking the agents' honest reports and simulating the optimal mechanism.

However, mechanisms in general are still hard to design, especially in scenarios where each agent reports multiple inputs. Generally speaking, traditional works usually try to provide closed-form implementations for a designed mechanism and theoretically prove its (approximated) optimality. This in some sense limits the studies on mechanism design for large-scale problems. In contrast, a rising research direction, using deep neural networks to represent complicated mechanisms, shows its potential to overcome this drawback. In this section, we call this direction "mechanism design with learning" and summarize recent works. We first introduce the most representative mechanism design problem, auction design, also known as mechanism design with money, which involves most related works. Then we introduce some learning applications on mechanism design without money. Finally, we introduce another combination of auction design and machine learning.

5.1 Machine Learning for Auction Design

We focus on the following setting of the auction design problem.

- A seller with a set of m items.
- A set of n bidders with independent private value $v_i : 2^m \to \mathbb{R}_{\geq 0}$, for $i = 1, 2, \ldots, n$.
- Each bidder i's value follows a known distribution $v_i \sim F_i$.
- An auction mechanism presented by an allocation rule $g(\cdot)$ and a payment rule $p(\cdot)$. This is, when bidder i bids b_i and other bidders bid b_{-i},
  - $g_i(b_i, b{-}i)$ denotes the probability (vector) that the items are allocated to bidder i;
  - $p_i(b_i, b{-}i)$ denotes the payment of bidder i;
  - then bidder i's utility is defined as $u_i(v_i, b_i, b_{-i}) = v_i^T \cdot g_i(b_i, b_{-i}) - p_i(b_i, b{-}i)$.
- Design a mechanism to maximize the expected revenue $\mathbb{E}_{v_1 \sim F_1, \ldots, v_n \sim F_n}[\sum_i p_i(v_i, v_{-i})]$, under the constraints of
  - Incentive compatible (IC): $\forall i, v_i, b_i, v_{-i}, u_i(v_i, v_i, v_{-i}) \geq u_i(v_i, b_i, v_{-i})$;
  - Individual rational (IR): $\forall i, v_i, v_{-i}, u_i(v_i, v_i, v_{-i}) \geq 0$.

For $m = 1$, Myerson [95] provides the revenue maximized auction mechanism, but even for $m = 2$, there is no complete analytical result. Many works that follow design optimal auctions for the multiple items setting with assumptions on fixed numbers of bidders (mostly two), or value distribution with finite support (two). Other works focus on designing mechanisms with constant approximation ratio compared with the optimal revenue.

Duetting et al. [36] first introduced deep neural networks to represent an auction mechanism. By doing so, they transferred the auction design problem into a deep learning task, which not only approximately recovers known optimal mechanisms, but also achieves more expected revenue than those approximate optimal mechanisms under corresponding settings. To be precise, the allocation rule $g(\cdot)$ and payment rule $p(\cdot)$ are parameterized by neural networks, denoted by $g^\theta(\cdot)$ and $p^\theta(\cdot)$. Specifically, $p_i^\theta = \alpha_i(v_i^T \cdot g_i^\theta)$ with $\alpha_i \in [0, 1]$, such that the IR constraint is directly satisfied. The IC constraint is transferred into a group of loss functions $rgt_i^\theta = \mathbb{E}_{v_1 \sim F_1, \ldots, v_n \sim F_n}[\max_{b_i} u_i^\theta(v_i, b_i, v_{-i}) - u_i^\theta(v_i, v_i, v_{-i})]$, namely each agent's regret for not misreporting. Then the IC constraint is satisfied if for all i, $rgt_i^\theta = 0$. Thus, the training process samples bidders' value from the known distributions, combines the negative expected revenue $-\mathbb{E}_{v_1 \sim F_1, \ldots, v_n \sim F_n}[\sum_i p_i^\theta(v_i, v_{-i})]$ and rgt_i^θ, and uses stochastic gradient descent (SGD) to update the neural networks' parameters θ.

Following them, Feng et al. [38] applied this approach to settings with budget constraints. Sakurai et al. [109] further included the false-name-proof constraints, that is, no bidder can benefit from manipulating fictitious identities to participate in the auctions, through a similar loss function as the IC constraints. Shen et al. [112] also adopted similar idea of the parameterizing mechanism by neural networks, but replaced the loss function related to IC constraints with another network representing the bidders' optimal actions with respect to the given mechanism. In other words, they do not leverage the revelation principle but search for the optimal auction mechanism in a larger parameterized space. Rahme et al. [105] further modeled the interaction between the mechanism network and the bidder network as a two-player game. On the other hand, Cai et al. [22] extended this line of learning assisted auction design to a dynamic setting. They considered running the auctions repeatedly, so that both the mechanism and the bidders may evolve along with time. However, to obtain a convincing performance, they restricted the design space of the auction mechanism as a second-price auction with reserve price, and used reinforcement learning to find the optimal reserve price for each time period. Recently, Liu et al. [78] applied the learning-based auction mechanism in a real-world scenario, to display ads auctions where the value distributions of bidders were assumed to have specific structures.

Other related works. The literature introduced above mainly considers learning-based approaches to design mechanisms. There are plenty of papers studying how many samples we need to learn an auction mechanism through theoretical analysis. Although this is more of a fundamental question, we will not survey the related works.

5.2 Machine Learning for Mechanism Design Without Money

Compared with the auction mechanisms represented by pairs of allocation rules and payment rules, many other mechanism design problems cannot involve payment rules. These problems are commonly referred to as mechanism design without money. Because there is no unified setting for them, we only introduce one recent work which leverages machine learning to augment the mechanism design problem.

Golowich et al. [47] considered the facility location problem, where the government asks agents to report their own locations and then find a location to build a facility beneficial to all agents. They used a similar idea as [38] to represent the "location rule" by a neural network, taking the agents' reports as input and the facility's location as output. The neural network is trained to minimize the total (weighted) distance between the output facility location and each agent. Each agent's incentive of misreporting (in order to make the facility's location as close to itself as possible) is also transformed into a loss function used in training.

5.3 Auction Design w.r.t. Learning Agents

Auction design can be regarded as a situation where the seller designs the rule of a game for buyers to play. The works mentioned above mainly use machine learning to augment the seller, while assuming buyers are rational, that is, buyers are able to find their optimal bids. To be precise, the revenue of the revenue-maximized auction mechanism is defined in the sense that each buyer's bid follows the Nash equilibrium, and the revelation principle further transforms the Nash equilibrium into truthful bidding. However, in practice, buyers may not be able to find the equilibrium. Thus, it is important to consider auction design with respect to non-rational buyers, especially learning agents, before applying an auction mechanism in real world.

A typical situation is that a seller repeatedly holds auctions for items of the same type, while independent buyers, probably with identical value distribution, use the no-regret learning algorithm to find bids achieving maximal long-term utility (minimizing regret). Nekipelov et al. [97] initialized the studies on this situation. They focused on second-price auction and showed how to estimate buyers' value from their bids by assuming they are all no-regret learners. Recently, Deng et al. [29] considered first-price auction, for which the Nash equilibrium has not been theoretically characterized, and showed that there exists a no-regret learning algorithm for buyers to obtain the Nash equilibrium.

6 Limitations and Challenges

We summarize some current limitations and challenges by incorporating ML for combinatorial optimization. We first discuss the differences between combinatorial optimization and pattern recognition, which induces several difficulties. Then we propose some properties of augmented algorithms that are required to be addressed in current researches.

6.1 Combinatorial Optimization v.s. Pattern Recognition

In the following, we summarize the difficulties of combinatorial optimization for incorporating ML when compared with pattern recognition.

1. **Not easy to construct a high-quality training dataset.** Nowadays, training an ML model with good performance usually requires a large training dataset, and hence, to obtain a high-quality dataset is an important problem. In pattern recognition, a data point usually consists of features and labels. To obtain training data, we have several ways that come with an acceptable cost, e.g., hiring people to mark labels. However, the label of a combinatorial optimization instance is usually an optimal solution. The cost of computing an optimal solution is expensive, especially for large scale instances, and it is unlikely that a human can mark this "label", as can be done in pattern recognition. Hence, it is a challenge to construct a high-quality dataset that consists of various instances with different scales.
2. **Sensitive to perturbations/noises.** In pattern recognition, an important requirement for an ML model is its robustness, i.e., ML models should not be sensitive to small perturbations or noises. To achieve this goal, we may slightly perturb training data while maintaining labels to increase the robustness of the learned ML models. However, combinatorial optimization problems may have a different property: The optimal solution can be very sensitive to small perturbations/noises. For instance, in the TSP, adding a single edge can make the optimal solution different; see Figure 4 for an example that adding an edge (A, D) changes half of the edges in the optimal solution. Another sensitive case is the SAT problem in which adding a clause may change the state from satisfiable to unsatisfiable. Also, adding one constraint to CSP or removing one tuple in a constraint relation of CSP may change an easy satisfiable problem into a hard one. This sensitivity of combinatorial optimization also makes it difficult to construct a high-quality training dataset because one cannot construct various instances via a small perturbation as can be done in pattern recognition. Consequently, we hope that the learned ML models can recognize this sensitivity, which is a different requirement from pattern recognition and also a big challenge.

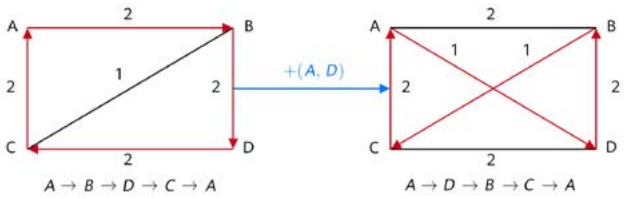


Figure 4 A traveling salesman instance that shows the sensitivity

6.2 Desired Properties of Augmented Algorithms

Despite recent progress in incorporating ML into combinatorial optimization, there are still many research challenges that need to be addressed for designing augmented algorithms. Challenges come from specific properties of combinatorial optimization, lack of understanding of theoretical/heuristic approaches, and practical needs.

1. **Scalability.** Due to the challenge in achieving training data of large scales, current ML models for combinatorial optimization usually train on small scale datasets, say at most 1000 variables. However, we expect to design augmented algorithms that also have a good performance on inputs with a larger size. Hence, we care about the scalability of the incorporated ML models, which has been listed as a future research direction in several recent surveys [114, 87, 19]. There have been some works [111, 68, 90] that attempt to scale to much larger instances. However, many existing augmented frameworks do not have good scalability: the quality of the solution drops quickly when the instance size increases [114].
2. **Adaptability.** In application scenarios, we usually have additional objectives or constraints compared to the classic combinatorial optimization problems. To reduce the cost of redesigning algorithms when the problem changes, we expect that the learned augmented framework can adapt to perturbations in the problem sets or other problems from the same family. There have been some initial works [96] that investigate the adaptability of ML models.
3. **Theoretical guarantees.** Traditional theoretical algorithms usually have provable guarantees, including *execution time, approximation ratio, and generalization bound*. It is a challenge to maintain these theoretical guarantees when incorporating ML algorithms, which can improve the interpretability of learned models.

Execution time is one of the most relevant factors along with the model performance, for practical implementation. Designed augmented algorithms are aimed at decreasing the running time while maintaining the quality of output solutions. There are some initial studies [17, 72] that achieve close solution performances with a faster time and focus on improving the efficiency of ML models. However, their framework can only handle 100 nodes, which is still quite far from being state-of-the-art.

Approximation ratio is to provide a theoretical guarantee for the quality of output solutions. One way to achieve an approximation guarantee is to incorporate approximation algorithms with ML methods, e.g., the vertex cover problem and the TSP problem [68]. However, existing combinations usually consider some basic theoretical/heuristic approaches, e.g., greedy algorithms or branch-and-bound. It is interesting to consider approximation algorithms that may be more complicated but can achieve a better approximation ratio. The combination of ML methods and better approximation algorithms may provide us with a better approximation guarantee.

Generalization reflects the ability of the learned models to perform well on unseen instances, for the models to be of practical interest. Traditional theoretical algorithms consider the worst-case bound and hence naturally satisfy generalization. However, the learned models of ML usually depend on the training dataset and generalization is an important problem of ML. It is interesting to investigate how to maintain generalization while incorporating ML methods.

Sample complexity is relevant to all the above factors. The number of samples usually decides the training time and the quality of learned models, which may affect the approximation ratio of the augmented algorithms. Also, if training data and testing data are drawn from the same distribution, the generalization bound usually relates to how well the training dataset estimates the underlying distribution, which heavily depends on the sample complexity.

References

[1] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1-2):203–218, 2000.

[2] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1253–1264. SIAM, 2011.

[3] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *International Conference on Machine Learning*, pages 345–355. PMLR, 2020.

[4] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In *34th Conference on Neural Information Processing Systems*. Curran Associates, Inc., 2020.

[5] Sanjeev Arora. Polynomial time approximation schemes for Euclidean travelling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[6] Ashwinkumar Badanidiyuru, Shahar Dobzinski, Hu Fu, Robert Kleinberg, Noam Nisan, and Tim Roughgarden. Sketching valuation functions. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1025–1035.

[7] Yunsheng Bai, Derek Xu, Alex Wang, Ken Gu, Xueqing Wu, Agustin Marinovic, Christopher Ro, Yizhou Sun, and Wei Wang. Glsearch: Maximum common subgraph detection via learning to search. *CoRR*, abs/2002.03129, 2020.

[8] Maria-Florina Balcan and Nicholas J. A. Harvey. Learning submodular functions. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol,*

UK, September 24-28, 2012. Proceedings, *Part II*, volume 7524 of *Lecture Notes in Computer Science*, pages 846–849. Springer, 2012.

[9] Eric Balkanski, Nicole Immorlica, and Yaron Singer. The importance of communities for learning to influence. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5862–5871, 2017.

[10] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The power of optimization from samples. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4017–4025, 2016.

[11] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The limitations of optimization from samples. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017,* pages 1016–1027. ACM, 2017.

[12] Eric Balkanski and Yaron Singer. Minimizing a submodular function from samples. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 814–822, 2017.

[13] Eric Balkanski and Yaron Singer. The sample complexity of optimizing a convex function. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017,* volume 65 of *Proceedings of Machine Learning Research*, pages 275–301. PMLR, 2017.

[14] Radu Baltean-Lugojan, Pierre Bonami, Ruth Misener, and Andrea Tramontani. Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks. *Technical Report, Imperial College, London*, 2018.

[15] Nikhil Bansal, Kedar Dhamdhere, Jochen Könemann, and Amitabh Sinha. Non-clairvoyant scheduling for minimizing mean slowdown. *Algorithmica*, 40(4):305–318, 2004.

[16] Luca Becchetti and Stefano Leonardi. Non-clairvoyant scheduling to minimize the average flow time on single and parallel machines. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 94–103, 2001.

[17] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[18] Alberto Bemporad and Vihangkumar V. Naik. A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control. *IFAC-PapersOnLine*, 51:412–417, 2018.

[19] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 2020.

[20] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

[21] Pierre Bonami, Andrea Lodi, and Giulia Zarpellon. Learning a classification of mixed-integer quadratic programming problems. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 595–604. Springer, 2018.

[22] Qingpeng Cai, Aris Filos-Ratsikas, Pingzhong Tang, and Yiwei Zhang. Reinforcement mechanism design for e-commerce. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1339–1348,

Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[23] Wei Chen, Xiaoming Sun, Jialin Zhang, and Zhijie Zhang. Optimization from structured samples for coverage functions. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event,* volume 119 of *Proceedings of Machine Learning Research*, pages 1715–1724. PMLR, 2020.

[24] Wei Chen, Xiaoming Sun, Jialin Zhang, and Zhijie Zhang. Network inference and influence maximization from samples. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, *ICML 2021, 18-24 July 2021, Virtual Event,* volume 139 of *Proceedings of Machine Learning Research*, pages 1707–1716. PMLR, 2021.

[25] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

[26] Graham Cormode and S Muthukrishnan. Space efficient mining of multigraph streams. In *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 271–282, 2005.

[27] Janos Csirik, David S Johnson, Claire Kenyon, James B Orlin, Peter W Shor, and Richard R Weber. On the sum-of-squares algorithm for bin packing. *Journal of the ACM (JACM)*, 53(1):1–65, 2006.

[28] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*, 13:339–347, 1951.

[29] Xiaotie Deng, Xinyan Hu, Tao Lin, and Weiqiang Zheng. Nash convergence of mean-based learning algorithms in first price auctions, 2021.

[30] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the TSP by policy gradient. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 170–181. Springer, 2018.

[31] Nikhil R Devanur and Thomas P Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, pages 71–78, 2009.

[32] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[33] Marco Dorigo and Gianni Di Caro. Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE, 1999.

[34] Lu Duan, Haoyuan Hu, Yu Qian, Yu Gong, Xiaodong Zhang, Jiangwen Wei, and Yinghui Xu. A multi-task selected learning approach for solving 3D flexible bin packing problem. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1386–1394, 2019.

[35] Zhijian Duan, Dinghuai Zhang, Wenhan Huang, Yali Du, Yaodong Yang, Jun Wang, and Xiaotie Deng. PAC learnability of approximate Nash equilibrium in bimatrix games, 2021.

[36] Paul Duetting, Zhe Feng, Harikrishna Narasimhan, David Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1706–1715. PMLR, 09–15 Jun 2019.

[37] Evgenii Borisovich Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Mathematics*, 4:627–629, 1963.

[38] Zhe Feng, Harikrishna Narasimhan, and David C. Parkes. Deep learning for revenue-optimal auctions with budgets. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 354–362, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[39] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[40] Merrill M Flood. The travelling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[41] Alex S Fukunaga and Richard E Korf. Bin completion algorithms for multicontainer packing, knapsack, and covering problems. *Journal of Artificial Intelligence Research*, 28:393–429, 2007.

[42] Fred Glover. Tabu search: part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[43] Fred Glover. Tabu search: part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[44] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of Combinatorial Optimization*, pages 2093–2229. Springer, 1998.

[45] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684, 2000.

[46] David E Goldberg. *Genetic Algorithms*. Pearson Education India, 2006.

[47] Noah Golowich, Harikrishna Narasimhan, and David C. Parkes. Deep learning for multi-facility location mechanism design. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 261–267. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[48] Avinatan Hassidim and Yaron Singer. Submodular optimization under noise. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017,* volume 65 of *Proceedings of Machine Learning Research*, pages 1069–1122. PMLR, 2017.

[49] He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[50] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210, 1962.

[51] Holger H Hoos. Automated algorithm configuration and parameter tuning. In *Autonomous Search*, pages 37–71. Springer, 2011.

[52] Andre Hottung, Shunji Tanaka, and Kevin Tierney. Deep learning assisted heuristic tree search for the container pre-marshalling problem. *Computers & Operations Research*, 113:104781, 2020.

[53] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2018.

[54] Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. *Journal of the ACM (JACM)*, 65(1):1–33, 2017.

[55] Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 531–540. IEEE, 2014.

[56] Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 285–294, 2021.

[57] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3): 307–323, 2006.

[58] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 202–208, 2005.

[59] Thathachar S Jayram and David P Woodruff. The data stream space complexity of cascaded norms. In

2009 50th Annual IEEE Symposium on Foundations of Computer Science, pages 765–774. IEEE, 2009.

[60] Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P Woodruff. Learning-augmented data stream algorithms. In *International Conference on Learning Representations*, 2019.

[61] Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 41–52, 2010.

[62] Daniel Karapetyan, Abraham P Punnen, and Andrew J Parkes. Markov chain methods for the bipartite boolean quadratic programming problem. *European* Journal of *Operational Research*, 260(2):494–506, 2017.

[63] Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[64] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[65] Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *Journal of Computer and System Sciences*, 81(8): 1665–1677, 2015.

[66] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European Symposium on Algorithms*, pages 589–600. Springer, 2013.

[67] Leonid Genrikhovich Khachiyan. A polynomial algorithm in linear programming. In *Doklady Akademii Nauk*. Russian Academy of Sciences, 1979.

[68] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

[69] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[70] Victor Klee and George J Minty. How good is the simplex algorithm. *Inequalities*, 3(3):159–175, 1972.

[71] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2018.

[72] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[73] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 508–520. Springer, 2009.

[74] Markus Kruber, Marco E Lübbecke, and Axel Parmentier. Learning when to use a decomposition. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 202–210. Springer, 2017.

[75] Eugene L Lawler. The travelling salesman problem: A guided tour of combinatorial optimization. *Wiley-Interscience Series in Discrete Mathematics*, 1985.

[76] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[77] Denis V Lindley. Dynamic programming and decision theory. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 10(1): 39–51, 1961.

[78] Xiangyu Liu, Chuan Yu, Zhilin Zhang, Zhenzhe Zheng, Yu Rong, Hongtao Lv, Da Huo, Yiqing Wang, Dagui Chen, Jian Xu, Fan Wu, Guihai Chen, and Xiaoqiang Zhu. Neural auction: End-to-end learning of auction mechanisms for e-commerce advertising. KDD '21, page

3354–3364, New York, NY, USA, 2021. Association for Computing Machinery.

[79] Yanli Liu, Chu-Min Li, Hua Jiang, and Kun He. A learning based branch and bound for maximum common subgraph related problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34: 2392–2399, 04 2020.

[80] Andrea Lodi and Giulia Zarpellon. On learning and branching: A survey. *Top*, 25(2):207–236, 2017.

[81] Michele Lombardi and Michela Milano. Boosting combinatorial problem modeling with machine learning. *arXiv preprint arXiv:1807.05517*, 2018.

[82] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of Metaheuristics*, pages 320–353. Springer, 2003.

[83] Thodoris Lykouris and Sergei Vassilvtiskii. Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pages 3296–3305. PMLR, 2018.

[84] Qiang Ma, Suwen Ge, Danyang He, Darshan Thaker, and Iddo Drori. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. *arXiv preprint arXiv:1911.04936*, 2019.

[85] Franco Mascia, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, and Thomas Stützle. Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. *Computers & Operations Research*, 51:190–199, 2014.

[86] Daniele Masti and Alberto Bemporad. Learning binary warm starts for multiparametric mixed-integer quadratic programming. pages 1494–1499, 06 2019.

[87] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *arXiv preprint arXiv:2003.03600*, 2020.

[88] Jesper W Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.

[89] Joseph SB Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[90] Akash Mittal, Anuj Dhawan, Sahil Manchanda, Sourav Medya, Sayan Ranu, and Ambuj Singh. Learning heuristics over large graphs via deep reinforcement learning. *arXiv preprint arXiv:1903.03332*, 2019.

[91] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *arXiv preprint arXiv:2006.09123*, 2020.

[92] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[93] Rajeev Motwani, Steven Phillips, and Eric Torng. Nonclairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994.

[94] Katta G Murty. Computational complexity of parametric linear programming. *Mathematical Programming*, 19(1):213–219, 1980.

[95] Roger B Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1) 58–73, 1981.

[96] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, pages 9839–9849, 2018.

[97] Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, page 1–18, New York, NY, USA, 2015. Association for Computing Machinery.

[98] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

[99] Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. In *Proceeding of the 34th International Conference*

on Machine Learning (ICML), volume 1050, page 22, 2017.

[100] Pekka Orponen and Heikki Mannila. On *Approximation Preserving Reductions: Complete Problems and Robust Measures*. Citeseer, 1987.

[101] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[102] Marcelo Prates, Pedro HC Avelar, Henrique Lemos, Luis C Lamb, and Moshe Y Vardi. Learning to solve NP-complete problems: A graph neural network for decision TSP. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4731–4738, 2019.

[103] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems*, pages 9661–9670, 2018.

[104] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[105] Jad Rahme, Samy Jelassi, and S. Matthew Weinberg. Auction learning as a two-player game. In *International Conference on Learning Representations*, 2021.

[106] Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1834–1845. SIAM, 2020.

[107] Nir Rosenfeld, Eric Balkanski, Amir Globerson, and Yaron Singer. Learning to optimize combinatorial functions. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research*, pages 4371–4380.

[108] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., USA, 2006.

[109] Yuko Sakurai, Satoshi Oyama, Mingyu Guo, and Makoto Yokoo. *Deep False-Name-Proof Auction Mechanisms*, pages 594–601. 10 2019.

[110] Ethan L Schreiber and Richard E Korf. Improved bin completion for optimal bin packing and number partitioning. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[111] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L Dill. Learning a SAT solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*, 2018.

[112] Weiran Shen, Pingzhong Tang, and Song Zuo. *Automated Mechanism Design via Neural Networks*, page 215–223. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2019.

[113] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms Why the simplex algorithm usually takes polynomial time. *Journal of the ACM* (JACM), 51(3):385–463, 2004.

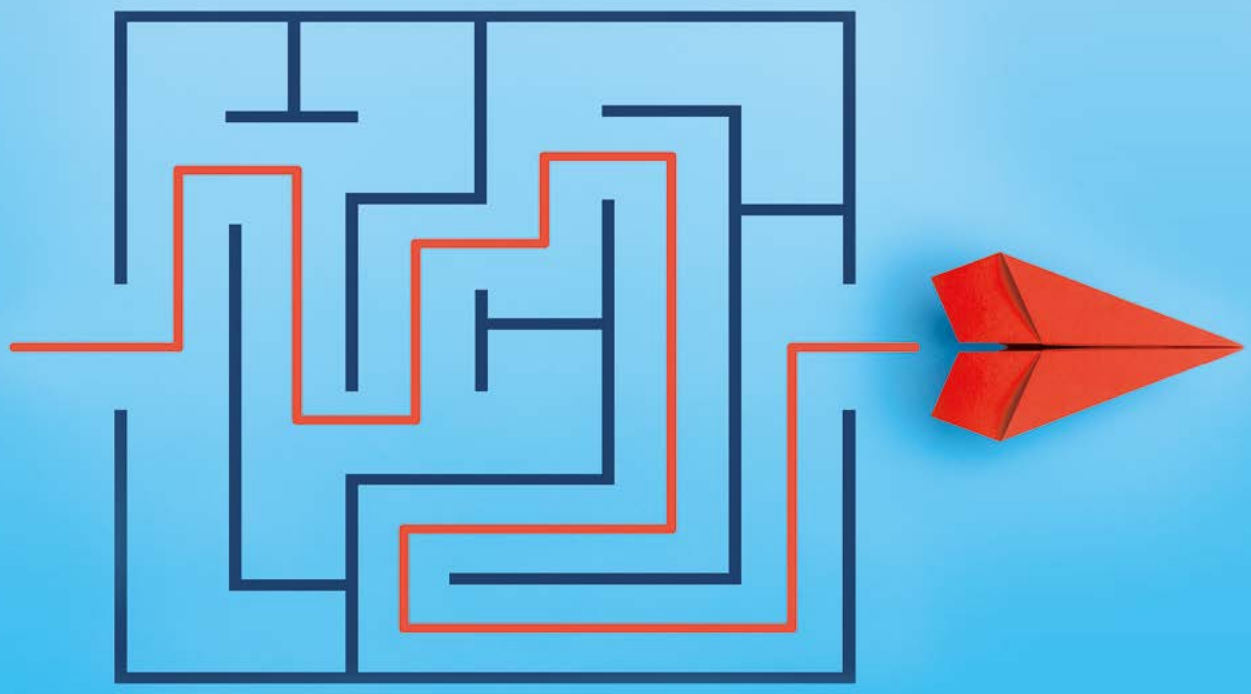
[114] Natalia Vesselinova, Rebecca Steinert, Daniel F Perez-Ramirez, and Magnus Boman. Learning combinatorial optimization on graphs: A survey with applications to networking. *arXiv preprint arXiv:2005.11081*, 2020.

[115] Alexander Wei. Better and simpler learning-augmented online caching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[116] Gerhard J Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization: Eureka, You Shrink!*, pages 185–207. Springer, 2003.

[117] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving the travelling salesman problem. *arXiv preprint arXiv:1912.05784*, 2019.

[118] Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, and Chu-Min Li. Combining reinforcement learning with Lin-Kernighan-Helsgaun algorithm for the traveling salesman problem. *CoRR*, abs/2012.04461, 2020.



Combining Random Walk with a Fitness Function for Boolean Satisfiability

Shaowei Cai, Tao Jiang
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences

Abstract

The satisfiability problem (SAT) plays a central role in computer science. While many heuristic algorithms can solve SAT, they are usually evaluated through experimentation, and theoretical analysis is rare. In this paper, we mainly focus on random walk (RW), which we improve by combining it with a fitness function. To demonstrate the performance of our improved version of RW, which we call RWF, we analyze and compare the expected runtime of RW, a typical evolution algorithm named (1+1) EA, and RWF in three SAT formulas. The results show that RWF can outperform RW and (1+1) EA in some instances.

Keywords

Boolean satisfiability, random walk, runtime analysis

1 Introduction

The Boolean satisfiability problem (SAT) is the problem to test whether there is an assignment to the variables that makes a given propositional formula true. It plays a central role in computer science. SAT is the first problem to be proven NP-complete, meaning that all problems in the complexity class NP are at most as difficult to solve as SAT. Besides the significant importance in theory, SAT algorithms are also used extensively in real-world applications, including in hardware verification, cryptography, resource allocation and planning.

In theory, SAT is a very challenging problem. To date, no efficient algorithm has been found, and no one can break the trivial bound $O^*(2^n)$ given by the simple brute force algorithm trying all 2^n assignments of the n variables. Indeed, the strong exponential time hypothesis (SETH) is the conjecture that the SAT problem cannot be solved in time $O^*(c^n)$ for some constant $c < 2$ [7]. Most theoretical algorithms have focused on the random k-SAT problem with a fixed k, where each clause consists of k random literals. Among these algorithms, a local search algorithm by Schöning [11] and the encoding-based algorithm called PPSZ [10] stand out as two typical examples of fast algorithms.

Local search has become one of the two main practical methods for solving SAT. The other is the conflict-driven clause learning (CDCL) method [2], which integrates powerful reasoning techniques and evolved from the backtracking DPLL algorithm. Typically, local search algorithms start from a complete assignment and flip the value of one variable in each step, aiming to find a solution. One key technique in local search for SAT is random walk (also known as focused random walk), which always selects a variable to flip from a randomly unsatisfied clause. This is the core idea of Schöning's local search algorithm, leading to an $O(poly(n)(2(1-1/k))^n)$ time complexity for k-SAT. Besides theoretical interest, RW has also been used in successful practical local search SAT solvers, such as WalkSAT [4], ProbSAT [1], and CCAnr [3]. However, there remain limited works on theoretical analyses of local search algorithms for SAT [5, 12].

Zhou et al. [12] compared the expected runtime of RW heuristic algorithms on some SAT instances, and in [5], the authors introduced a random local search algorithm based on Hamming balls. Conversely, a great deal of effort has been devoted to analyzing evolutionary algorithms (EAs) for SAT, for example, local search algorithms with particular operations such as mutation. Zhou et al. compared the expected runtime of a simple random local search with a simple EA denoted as (1+1) EA on a few crafted instances [12]. EAs are guided by a fitness function, which is missing in RW.

In this paper, we focus on a popular stochastic local search (SLS) method named RW, which has been used in many successful SLS solvers for SAT (e.g., WalkSAT [4], ProbSAT [1], and CCAnr [3]). We propose to improve RW by combining it with a fitness function typically used in EAs, leading to an improved version called RW with Fitness evaluation (RWF).

We compare RW, (1+1) EA, RWF with three SAT formulas. Two of them taken from previous literature, and the last is proposed in this work. We analyze the worst case or average case complexity for the instances with Markov chains used mainly. The runtime analysis of the three heuristic algorithms can be exponential time or polynomial time. In other words, they have their own advantages and disadvantages in solving different SAT instances.

The remainder of the paper is organized as follows. Section 2 introduces some prerequisite knowledge about the SAT problem, absorbing Markov chains, and two vector norms. In Section 3.1, we introduce these three heuristic algorithms, and in Section 3.2, we analyze and compare the expected time complexity of RW and RWF on two 2-SAT instances. Then in Section 3.3, we analyze and compare the expected time complexity of the three heuristic algorithms on a 3-SAT instance.

2 Preliminaries

2.1 Boolean Satisfiability Problem (SAT)

In Boolean logic, a literal is a variable or its negation (i.e., x or $\overline{x}$). A clause is a disjunction of literals (e.g., $x \vee y$ is a clause with two literals). A formula in conjunctive normal form (CNF) is a conjunction of clauses. A k-SAT instance is a CNF formula where each clause has exactly k literals. SAT is the problem of whether the variables of a given Boolean formula can be assigned values to make the formula to true.

2.2 The Absorbing Markov Chain

Let $\{X_t\}$ be a discrete absorbing Markov chain with finite state space S. T is the transient state set, and A=S–T is the absorbing set. Let $|\cdot|$ denote the cardinality of the set. Assuming that $|T| = t$ and $|A| = r$, then the transition matrix is:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{R} & \mathbf{T} \end{pmatrix},$$

where $\mathbf{I}$ is the matrix of r-by-r, $\mathbf{R}$ is a nonzero matrix of t-by-r, and $\mathbf{T}$ is a nonzero matrix of t-by-t.

Theorem 2.2.1 Given that the absorbing Markov chain starts in state i, let m_i be the expected number of steps before it hits the absorbing state. We denote $\mathbf{m} = [m_i]_{i\in T}$. Then

$$\mathbf{m} = (\mathbf{I}_{t*t} - \mathbf{T})^{-1}\mathbf{1}.$$

Proof: See [8].

2.3 Two Vector Norms

- **Average vector norm:** For vector $\mathbf{X} = [x_i]_{i\in S}$, let $P(X_0 = i)$ be the initial distribution. Then

$$||\mathbf{m}||_1 = \sum_{i\in S} P(X_0 = i)m_i$$

- **The maximum vector norm:**

$$||\mathbf{m}||_\infty = \max_{i\in S}\{m_i\}$$

Norm $||\mathbf{m}||_1$ presents average case in the time complexity analysis, while $||\mathbf{m}||_\infty$ presents the worst case.

3 Comparison of Three Heuristic Algorithms

3.1 Introduction

Evolutionary algorithms (EAs) have been applied to SAT problems and typically use a fitness function to guide the search process. In the MaxSAT problem, a canonical fitness function fit is the number of satisfied clauses. Specifically, given a CNF formula consisting of clauses $c_1, \cdots, c_m$, for any assignment x to the variables,

$$fit(x) = c_1(x) + c_2(x) + \cdots + c_m(x), \quad (1)$$

where, for $1 \le i \le m$, $c_i(x) = 1$ if c_i is satisfied by x; otherwise, $c_i(x) = 0$.

Throughout this paper, we will use the mentioned fitness function.

RW is one of the basic incomplete algorithms introduced by Papadimitiou [9]. It randomly selects an unsatisfied clause in the first step, and randomly flips a variable in the selected clause in the second step (see Algorithm 1).

Algorithm 1 Random walk (RW)

1: Select an initial bit string x at random;
2: **while** the formula is not satisfied **do**
3: Randomly select an unsatisfied clause c;
4: Randomly select a variable x_i in c and flip its value
5: **end while**

The design of EAs is inspired from modeling the processes of natural selection and genetic evolution. Here we consider a simple EA using mutation and selection approaches with a population size of 1 denoted as (1 + 1) EA [6]. (1 + 1) EA is a simple but effective random hill-climbing EA and generally uses two kinds of mutation, called local mutation and global mutation.

- Local mutation randomly chooses a bit $x_i(1 \le i \le n)$ from the individual $x = (x_1 \cdots x_n) \in \{0,1\}^n$ and flips it.
- Global mutation flips each bit of individual $x = (x_1 \cdots x_n) \in \{0,1\}^n$ independently with the probability of $1/n$. The expected number of bit flips for the global mutation is 1

Throughout this paper, we use the local (1+1) EA.

Algorithm 2 Local (1+1) EA

1: Select an initial bit string x at random;
2: **while** the formula is not satisfied **do**
3: Randomly choose a bit x_i from $x = (x_1, \cdots, x_n)$ and flip it, producing a solution y;
4: if fitness(y)>fitness(x), x:=y
5: **end while**

The following algorithm, which combines RW and a fitness function, is the focus of our analysis. In each step, we use a fitness value to guide the search process.

Algorithm 3 RW with Fitness evaluation (RWF)

1: Select an initial bit string x at random;
2: **while** the formula is not satisfied **do**
3: Randomly select an unsatisfied clause c;
4: Randomly select a variable x_i in c and flip its value, producing a solution y;
5: if fitness(y)>fitness(x), x:=y
6: **end while**

3.2 Comparison between RW and RWF

In order to obtain a theoretical understanding of RW and RWF, we analyze their time complexity (both average and worst) on two SAT instances.

Instance 3.2.1 For $x = (x_1 \cdots x_n) \in \{0,1\}^n$, the SAT instance $\psi_1(x)$ is defined as

$$\psi_1(x) = (x_1 \vee \overline{x}_2) \wedge \cdots (x_1 \vee \overline{x}_n) \wedge (\overline{x}_1 \vee x_2) \wedge \cdots (\overline{x}_1 \vee x_n).$$

The satisfying assignments are $(0 \cdots 0)$ and $(1 \cdots 1)$.

Proposition 3.2.1 For SAT instance ψ_1, the expected runtime of RW is $||\mathbf{m}||_1 = \Theta(n^2)$.

Proof: See [12].

Proposition 3.2.2 For SAT instance ψ_1, the expected runtime of RWF is $||\mathbf{m}||_1 = O(n)$.

Proof: Throughout the rest of this paper, we will use $|x|$ to denote the number of 1 in x.

According to Eq.(1), the fitness function of ψ_1 is

$$fit_{\psi 1}(x) = \begin{cases} 2(n-1) - |x|, & x = (0 * \cdots *), \\ n - 1 + |x| - 1, & x = (1 * \cdots *). \end{cases}$$

The fitness function is completely determined by the x_1 and $|x|$. Specifically, it decreases monotonically with $|x|$ if $x = (0 * \cdots *)$; otherwise, it increases monotonically. The fitness function $fit_{\psi_1}(x)$ with $n = 20$ is shown in Figure 1.

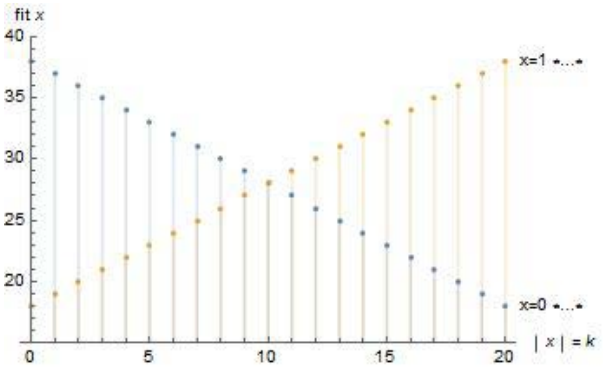


Figure 1 The fitness function of ψ_1 with $n = 20$

For simplicity, we assume that n is even.

We further divide the search space S into $2n$ subspaces:

$$S_{0,k} = \{x | x = (0 * \cdots *), |x| = k\} \quad (k = 0, \ldots, n-1),$$
$$S_{1,k} = \{x | x = (1 * \cdots *), |x| = k\} \quad (k = 1, \ldots, n).$$

Let $X_t \in \{0,1\}^n \, (t = 0, 1, \ldots)$ be the random variable describing the state of RWF solving SAT instance ψ_1 at time t. In this case, the transition probabilities can be described as follows.

When $k = 0$, we have:

$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = 1.$$

When $1 \leq k \leq \frac{n}{2} - 1$:

For $X_t \in S_{0,k}$, the unsatisfied clauses have the form of $x_1 \vee \overline{x}_i$. If the algorithm leads x_1 to 1, the fitness function will decrease. Consequently, the mutation will not remain:

$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = \frac{1}{2},$$
$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{0,k}) = \frac{1}{2}.$$

When $\frac{n}{2} \leq k \leq n - 1$:

For $X_t \in S_{0,k}$, the unsatisfied clauses have the form of $x_1 \vee \overline{x}_i$. If the algorithm leads x_1 to 1, the fitness function will increase. Consequently, the mutation will remain:

$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{0,k}) = \frac{1}{2},$$
$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{0,k}) = \frac{1}{2}.$$

Similarly, we have the following results.

When $k = n$:

$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = 1.$$

When $\frac{n}{2} + 1 \leq k \leq n - 1$:

$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = \frac{1}{2},$$
$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{1,k}) = \frac{1}{2}.$$

When $1 \leq k \leq \frac{n}{2}$:

$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{1,k}) = \frac{1}{2},$$
$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{1,k}) = \frac{1}{2}.$$

If we introduce an auxiliary homogeneous Markov chain $Z_t \quad (t = 0, 1, \ldots)$ with the state space $\{z_{0,0}, z_{0,1}, \ldots, z_{0,n-1}; z_{1,1}, z_{1,2}, \ldots, z_{1,n}\}$, the transition probabilities are defined by

$$P(Z_{t+1} = z_{v,h} | Z_t = z_{u,k}) = P(X_{t+1} \in S_{v,h} | X_t \in S_{u,k})$$

where $u, v \in 0, 1$, and $h, k \in 0, \ldots, n$.

In this case, Z_t is an absorbing Markov chain with the absorbing state $z_{0,0}$ and $z_{1,n}$. For any $x \in S_{u,k}(u \in 0,1,k \in 0,\ldots,n)$, the mean first hitting time m_x equals $m_{z_{u,k}}$.

According to the absorbing Markov chain theory, the mean first hitting time of stochastic process Z_t is given by

$$\begin{cases} m_{0,k} = 2k & 1 \le k \le \frac{n}{2} - 1, \\ m_{0,k} = \frac{1}{2}(m_{0,k-1}+1) + \frac{1}{2}(m_{1,k+1}+1) & \frac{n}{2} \le k \le n-1. \\ \\ m_{1,k} = 2(n-k) & \frac{n}{2}+1 \le k \le n-1 \\ m_{1,k}, = \frac{1}{2}(m_{0,k-1}+1) + \frac{1}{2}(m_{1,k+1}+1) & 1 \le k \le \frac{n}{2}. \end{cases}$$

In the following, we prove

$$m_{0,k} \le 2k \quad \text{where} \quad \frac{n}{2} \le k \le n-1.$$

When $k = \frac{n}{2}$, we obtain

$$\begin{aligned} m_{0,\frac{n}{2}} &= \frac{1}{2}(m_{0,\frac{n}{2}-1}+1) + \frac{1}{2}(m_{1,\frac{n}{2}+1}+1) \\ &= \frac{1}{2}(2(\frac{n}{2}-1)+1) + \frac{1}{2}(2(\frac{n}{2}-1)+1) \\ &= n-1. \end{aligned}$$

As such $m_{0,k} \le 2k$ holds for $k = \frac{n}{2}$. Assuming this is true for $k \ge \frac{n}{2}$ (i.e., $m_{0,k} \le 2k$), we have:

$$\begin{aligned} m_{0,k+1} &= \frac{1}{2}(m_{0,k}+1) + \frac{1}{2}(m_{1,k+2}+1) \\ &= \frac{1}{2}m_{0,k} + n - k - 1. \end{aligned}$$

Therefore,

$$m_{0,k+1} \le k + n - k - 1 \le 2(k+1) \quad k \ge \frac{n}{2}.$$

We can obtain a similar conclusion for $m_{1,k}$. We summarize them as follows:

$$\begin{cases} m_{0,k} \le 2k & 0 \le k \le n-1, \\ m_{1,k} \le 2(n-k) & 1 \le k \le n. \end{cases}$$

Therefore, $||\mathbf{m}||_1 = O(n)$.

For SAT instance ψ_1, the $O(n)$ expected runtime bound of RWF is better than the $O(n^2)$ expected runtime bound of RW. However, the opposite situation occurs under certain conditions for SAT instance ψ_2.

Instance 3.2.2 The SAT instance ψ_2 has the following clauses:

$$x_i, \quad x_i \vee \overline{x}_j \quad \text{where} \quad i,j \in \{1,\ldots,n\}, \quad i \ne j.$$

The only satisfying assignment of SAT instance ψ_2 is $(1,\cdots,1)$.

Proposition 3.2.3 For SAT instance ψ_2, the expected runtime of RW is $||\mathbf{m}||_\infty = \Omega(n^2)$.

Proof: See [12].

Proposition 3.2.4 For SAT instance ψ_2, the expected runtime of RWF is

$$m_i = \begin{cases} +\infty, & i < \frac{n-1}{2}, \\ \Theta(n), & i \ge \frac{n-1}{2}. \end{cases}$$

Proof: First, we divide the search space into $n+1$ subspaces:

$$S_i = \{x||x| = i, x \in \{0,1\}^n\} \qquad i = 0,1,\ldots,n.$$

Here, the satisfying set is S_n.

For $|x| = k$, the fitness function of SAT instance ψ_2 is given as

$$fit_{\psi_2}(x) = k^2 + k(1-n) + n(n-1).$$

It is a polynomial in k of degree 2. If the initial selected solution of RWF is $x(|x| < \frac{n-1}{2})$, according to the process of RWF, it will reach the local optimum $(0,\ldots,0)$, rather than the global optimum $(1,\ldots,1)$. When $k < \frac{n-1}{2}$, the RWF starting from S_k will never reach the satisfying assignment $(1,\cdots,1)$, because the fitness decreases as $|x|$ increase.

When $k \ge \frac{n-1}{2}$, we have:

$$\begin{aligned} P(X_{t+1} \in S_k | X_t \in S_k) &= \frac{1}{2}\frac{k}{1+k}, \\ P(X_{t+1} \in S_{k+1} | X_t \in S_k) &= 1 - \frac{1}{2}\frac{k}{1+k}. \end{aligned}$$

By the definition of expectation:

$$n - i \le m_i = \sum_{k=i}^{n-1} \frac{2k+2}{k+2} \le 2(n-i) \qquad \text{where} \quad i \ge \frac{n-1}{2}.$$

Thus, we complete the proof.

Contrary to the prior result that the expected runtime of RWF for SAT instance ψ_1 is better than that of RW, here we demonstrate that for SAT instance ψ_2, the expected runtime of RW is better than that of RWF.

We note that the analysis of RW for SAT is highly dependent on the fitness function. Because of the obstacles involved in writing out the expression of the fitness function, analyzing

the expected runtime of RWF for a general SAT problem is difficult. In the following, we construct an SAT instance $\psi_3(x)$, for which we analyze the expected runtime of RWF without writing the fitness function expression.

3.3 Comparison of RW, Local (1+1) EA, and RWF

Instance 3.3.1 For $x = (x_1 \cdots x_n) \in \{0,1\}^n$, the SAT instance $\psi_3(x)$ is formed by the following clauses:

$$(x_1 \vee \overline{x}_2), \cdots (x_1 \vee \overline{x}_n), (\overline{x}_1 \vee x_2) \cdots (\overline{x}_1 \vee x_n),$$

and

$$(x_1 \vee \overline{x}_2), \cdots (x_1 \vee \overline{x}_n), (\overline{x}_1 \vee x_2) \cdots (\overline{x}_1 \vee x_n),$$

and

$$(x_1 \vee \overline{x}_2 \vee x_3), (x_1 \vee x_2 \vee \overline{x}_3), (x_1 \vee \overline{x}_4 \vee x_5), (x_1 \vee x_4 \vee \overline{x}_5) \cdots (x_1 \vee \overline{x}_{n-1} \vee x_n), (x_1 \vee x_{n-1} \vee \overline{x}_n),$$

and

$$(\overline{x}_1 \vee \overline{x}_2 \vee x_3), (\overline{x}_1 \vee x_2 \vee \overline{x}_3), (\overline{x}_1 \vee \overline{x}_4 \vee x_5), (\overline{x}_1 \vee x_4 \vee \overline{x}_5) \ldots (\overline{x}_1 \vee \overline{x}_{n-1} \vee x_n), (\overline{x}_1 \vee x_{n-1} \vee \overline{x}_n).$$

The satisfying assignments are $(0 \cdots 0)$ and $(1 \cdots 1)$.

Note that there are two identical clauses with two literals. The following results are easy to obtain.

Lemma 3.3.1 For any assignment $x \in \{0,1\}^n$, there are twice as many unsatisfied clauses with two literals as there are unsatisfied clauses with three literals.

Lemma 3.3.2 When x_1 flips from 1 to 0 or from 0 to 1, it does not change the number of unsatisfied clauses with three literals.

Proposition 3.3.1 For SAT instance $\psi_3(x)$, the expected runtime of RW is $||\mathbf{m}||_\infty = O(2^n)$.

Proof: Let $S = \{0,1\}^n$ be the search space, S^* be the satisfying assignment set, $d(x) = \min_{y \in S^*} \{H(x,y)\}$ denote the Hamming distance between a point $x \in S$ and the set S^*, and $D_i = \{x \in S | d(x) = i\}, i = 0, 1, \cdots, n$. Then the search space S is partitioned into $n+1$ subspaces: $S = \bigcup_{i=0}^{n} D_i$

The probability that RW transfers x to some string $y \in D_{i-1}$ is at least $\frac{1}{3}$, and the probability that RW transfers x to some string $y \in D_{i+1}$ is at most $\frac{2}{3}$.

We construct an auxiliary homogeneous chain with the transition matrix

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

According to the absorbing Markov chain, we have

$$m_n = 2(n-1) + 6(2^{n-2} + n - 3)$$

Hence

$$||\mathbf{m}||_\infty = O(2^n)$$

The result is probably not tight because the bound for k-SAT is $O((k-1)^n)$.

Proposition 3.3.2 For SAT instance $\psi_3(x)$, the expected runtime of the local (1+1) EA is $||\mathbf{m}||_\infty = \Omega(n \ln n)$.

Proof: We divide the search space S into $2n$ subspaces:

$$S_{0,k} = \{x | x = (0 * \cdots *), |x| = k\} \quad (k = 0, \cdots, n-1),$$
$$S_{1,k} = \{x | x = (1 * \cdots *), |x| = k\} \quad (k = 1, \cdots, n).$$

Let $X_t \in \{0,1\}^n \, (t = 0, 1, \dots)$ be the random variable describing the state of local (1+1) EA solving SAT instance ψ_3 at time t. The transition probabilities can be described as follows.

When $k = 0$, we have:

$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = 1.$$

When $1 \le k \le \frac{n-1}{2}$, we have:

For $X_t \in S_{0,k}$, if the algorithm flips x_i from 0 to 1, the fitness function will decrease. Consequently, the mutation will not remain.

$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = 1 - \frac{k}{n},$$
$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{0,k}) = \frac{k}{n}.$$

When $\frac{n+1}{2} \le k \le n-1$, we have:

For $X_t \in S_{0,k}$, if the algorithm flips x_1 to 1, the fitness function will increase. Consequently, the mutation will remain:

$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{0,k}) = \frac{k}{n},$$
$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = 1 - \frac{k+1}{n},$$
$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{0,k}) = \frac{1}{n}.$$

Similarly, when $k = n$, we have:

$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = 1.$$

When $\frac{n+1}{2} \le k \le n$, we have:

For $X_t \in S_{1,k}$, if the algorithm flips x_1 from 1 to 0, the fitness function will decrease. Consequently, the change will not remain.

$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = 1 - \frac{n-k}{n},$$
$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{1,k}) = \frac{n-k}{n}.$$

When $1 \le k \le \frac{n-1}{2}$, we have:

For $X_t \in S_{1,k}$, if the algorithm leads x_1 to 0, the fitness function will increase. Consequently, the change will remain.

$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{1,k}) = \frac{n-k}{n},$$
$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = 1 - \frac{n-k+1}{n},$$
$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{1,k}) = \frac{1}{n}.$$

Here we introduce an auxiliary homogeneous Markov chain $Z_t \quad (t = 0, 1, \dots)$ with the state space $\{z_{0,0}, z_{0,1}, \dots, z_{0,n-1}; z_{1,1}, z_{1,2}, \dots, z_{1,n}\}$. The transition probabilities are defined by

$$P(Z_{t+1} = z_{v,h} | Z_t = z_{u,k}) = P(X_{t+1} \in S_{v,h} | X_t \in S_{u,k})$$

where $u, v \in 0, 1$, and $h, k \in 0, \dots, n$.

In this case, Z_t is an absorbing Markov chain with the absorbing state $z_{0,0}$ and $z_{1,n}$. For any $x \in S_{u,k} (u \in 0, 1, k \in 0, \dots, n)$, the mean first hitting time m_x equals $m_{z_{u,k}}$.

According to the absorbing Markov chain theory, the mean first hitting time of stochastic process Z_t is given by

$$\begin{cases} m_{0,k} = n(1 + \cdots + \frac{1}{k}) & 1 \le k = \frac{n-1}{2}, \\ m_{1,k} = n(1 + \cdots + \frac{1}{n-k}) & \frac{n+1}{2} = k \le n-1, \end{cases}$$

This means that $||\mathbf{m}||_\infty = \Omega(n \ln n)$.

Proposition 3.3.3 For SAT instance $\psi_3(x)$, the expected runtime of RWF is $||\mathbf{m}||_1 = O(n)$.

Proof: Let $S = \{0,1\}^n$ be the search space.

We divide the search space S into $2n$ subspaces:

$$S_{0,k} = \{x | x = (0 * \cdots *), |x| = k\} \quad (k = 0, \cdots, n-1),$$
$$S_{1,k} = \{x | x = (1 * \cdots *), |x| = k\} \quad (k = 1, \cdots, n).$$

We define $q_{i,k}$ as the probability that RWF chooses an unsatisfied clause with two literals in the first step when the current search point is in $S_{i,k}$.

According to Lemma 3.3.1, $q_{i,k} \ge \frac{2}{3}$.

Let $X_t \in \{0,1\}^n \, (t = 0, 1, \dots)$ be the random variable describing the state of RWF solving SAT instance ψ_3 at time t. The transition probabilities can be described as follows.

When $k = 0$, we have:

$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = 1.$$

When $1 \leq k \leq \frac{n-1}{2}$, we have:

For $X_t \in S_{0,k}$, if the algorithm leads x_1 to 1, the fitness function will decrease. Consequently, the mutation will not remain.

$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = \frac{1}{2} q_{0,k} + \frac{2}{3}(1 - q_{0,k}),$$
$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{0,k}) = \frac{1}{2} q_{0,k} + \frac{1}{3}(1 - q_{0,k}) \geq \frac{1}{3}.$$

When $\frac{n+1}{2} \leq k \leq n-1$, we have:

For $X_t \in S_{0,k}$, if the algorithm leads x_1 to 1, the fitness function will increase. Consequently, the mutation will remain.

$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{0,k}) = \frac{1}{2} q_{0,k} + \frac{1}{3}(1 - q_{0,k}),$$
$$P(X_{t+1} \in S_{0,k} | X_t \in S_{0,k}) = \frac{1}{3}(1 - q_{0,k}),$$
$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{0,k}) = \frac{1}{2} q_{0,k} + \frac{1}{3}(1 - q_{0,k}).$$

Similarly, when $k = n$, we have:

$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = 1.$$

When $\frac{n+1}{2} \leq k \leq n$, we have:

For $X_t \in S_{1,k}$, if the algorithm leads x_1 to 0, the fitness function will decrease. Consequently, the mutation will not remain.

$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = \frac{1}{2} q_{1,k} + \frac{2}{3}(1 - q_{1,k}).$$
$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{1,k}) = \frac{1}{2} q_{1,k} + \frac{1}{3}(1 - q_{1,k}) \geq \frac{1}{3}.$$

When $1 \leq k \leq \frac{n-1}{2}$, we have:

For $X_t \in S_{1,k}$, if the algorithm leads x_1 to 0, the fitness function will increase. Consequently, the mutation will remain.

$$P(X_{t+1} \in S_{1,k+1} | X_t \in S_{1,k}) = \frac{1}{2} q_{1,k} + \frac{1}{3}(1 - q_{1,k}).$$
$$P(X_{t+1} \in S_{1,k} | X_t \in S_{1,k}) = \frac{1}{3}(1 - q_{1,k}).$$
$$P(X_{t+1} \in S_{0,k-1} | X_t \in S_{1,k}) = \frac{1}{2} q_{1,k} + \frac{1}{3}(1 - q_{1,k}).$$

Here we introduce an auxiliary homogeneous Markov chain $Z_t \quad (t = 0, 1, \dots)$ with the state space $\{z_{0,0}, z_{0,1}, \dots, z_{0,n-1}; z_{1,1}, z_{1,2}, \dots, z_{1,n}\}$. The transition probabilities are defined by

$$P(Z_{t+1} = z_{v,h} | Z_t = z_{u,k}) = P(X_{t+1} \in S_{v,h} | X_t \in S_{u,k})$$

where $u, v \in 0, 1$, and $h, k \in 0, \dots, n$.

In this case, Z_t is an absorbing Markov chain with the absorbing state $z_{0,0}$ and $z_{1,n}$, and for any $x \in S_{u,k} (u \in 0, 1, k \in 0, \dots, n)$. The mean first hitting time m_x equals $m_{z_{u,k}}$.

According to the absorbing Markov chain theory, the mean first hitting time of stochastic process Z_t is given by

$$\begin{cases} m_{0,k} \leq 3k & 1 \leq k \leq \frac{n-1}{2}, \\ m_{0,k} = \frac{1}{2}(m_{0,k-1} + 1) + \frac{1}{2}(m_{1,k+1} + 1) + \frac{1 - q_{0,k}}{2 + q_{0,k}} & \frac{n+1}{2} \leq k \leq n-1. \\ \\ m_{1,k} \leq 3(n-k) & \frac{n+1}{2} \leq k \leq n-1, \\ m_{1,k} = \frac{1}{2}(m_{0,k-1} + 1) + \frac{1}{2}(m_{1,k+1} + 1) + \frac{1 - q_{1,k}}{2 + q_{1,k}} & 1 \leq k \leq \frac{n-1}{2}. \end{cases}$$

In the following, we prove

$$m_{0,k} \leq 3k \quad \text{where} \quad \frac{n+1}{2} \leq k \leq n-1.$$

When $k = \frac{n+1}{2}$, we obtain:

$$\begin{aligned} m_{0,\frac{n+1}{2}} &= \frac{1}{2}(m_{0,\frac{n-1}{2}} + 1) + \frac{1}{2}(m_{1,\frac{n+3}{2}} + 1) + \frac{1 - q_{0,\frac{n+1}{2}}}{2 + q_{0,\frac{n+1}{2}}} \\ &= \frac{1}{2}(3(\frac{n-1}{2}) + 1) + \frac{1}{2}(3(\frac{n-3}{2}) + 1) + \frac{1 - q_{0,\frac{n+1}{2}}}{2 + q_{0,\frac{n+1}{2}}} \\ &= \frac{3n}{2} + \frac{1 - q_{0,\frac{n+1}{2}}}{2 + q_{0,\frac{n+1}{2}}} \\ &\leq 3\frac{n+1}{2}. \end{aligned}$$

As such, $m_{0,k} \leq 3k$ holds for $k = \frac{n+1}{2}$.

Assuming this is true for $k \geq \frac{n+1}{2}$ (i.e., $m_{0,k} \leq 3k$), we have

$$\begin{aligned} m_{0,k+1} &= \frac{1}{2}(m_{0,k} + 1) + \frac{1}{2}(m_{1,k+2} + 1) + \frac{1 - q_{0,\frac{n+1}{2}}}{2 + q_{0,\frac{n+1}{2}}} \\ &\leq \frac{1}{2}m_{0,k} + \frac{3}{2}n - \frac{3}{2}k - 2 + \frac{1 - q_{0,\frac{n+1}{2}}}{2 + q_{0,\frac{n+1}{2}}} \\ &\leq \frac{3n}{2} - 2 + \frac{1 - q_{0,\frac{n+1}{2}}}{2 + q_{0,\frac{n+1}{2}}} \\ &\leq 3(k+1). \end{aligned}$$

Therefore,

$$m_{0,k} \leq 3k \quad \text{where} \quad 1 \leq k \leq n-1.$$

We can obtain a similar conclusion for $m_{1,k}$. We summarize them as follows:

$$\begin{cases} m_{0,k} \leq 3k & 0 \leq k \leq n-1 \\ m_{1,k} \leq 3(n-k) & 1 \leq k \leq n \end{cases}$$

Therefore, $||\mathbf{m}||_1 = O(n)$.

4 Conclusion

We use two 2-SAT instances, construct a 3-SAT instance, and provide analytic comparisons among RW, local (1 + 1) EA, and RWF on these instances. We show that these heuristic algorithms have their own advantages and disadvantages in solving these three SAT instances and that their expected runtime ranges from a polynomial time to an exponential time. Our analysis provides insight into the runtime behavior among these heuristic algorithms. RWF performs better in a situation where no local optimum is present (because a local optimum causes the worst expected runtime to infinity).

We summarize our comparison of the three SAT instances in the following table.

	Local (1+1) EA	RW	RWF
ψ_1	$\Theta(n \ln n)$	$\Theta(n^2)$	$O(n)$
ψ_2	$+\infty$	$\Theta(n^2)$	$+\infty$
ψ_3	$\Omega(n \ln n)$	$O(2^n)$	$O(n)$

Acknowledgments

We would like to thank Professor Xingwu Liu for pointing out a mistake in previous proof and providing helpful comments and suggestions.

References

[1] Adrian Balint and Uwe Schöning. "Choosing probability distributions for stochastic local search and the role of make versus break". In: *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing*. Berlin, Heidelberg, 2012, pp. 16–29.

[2] Armin Biere et al. "Conflict-driven clause learning SAT solvers". In: *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications* (2009), pp. 131–153.

[3] Shaowei Cai, Chuan Luo, and Kaile Su. "CCAnr: a configuration checking based local search solver for non-random satisfiability ". In: *SAT*. 2015.

[4] Amin Coja-Oghlan et al. "On smoothed CNF formulas and the WalkSAT algorithm". In: *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2009, pp. 451–460.

[5] Evgeny Dantsin, Edward A Hirsch, and Alexander Wolpert. "Algorithms for SAT based on search in Hamming balls". In: *Annual Symposium on Theoretical Aspects of Computer Science*. 2004, pp. 141–151.

[6] Stefan Droste, Thomas Jansen, and Ingo Wegener. "On the analysis of the (1 + 1) evolutionary algorithm". In: *Theoretical Computer Science* 276.1-2 (2002), pp. 51–81.

[7] Russell Impagliazzo and Ramamohan Paturi. "On the complexity of k-SAT". In: *J. Comput. Syst. Sci.* 62.2 (2001), pp. 367–375.

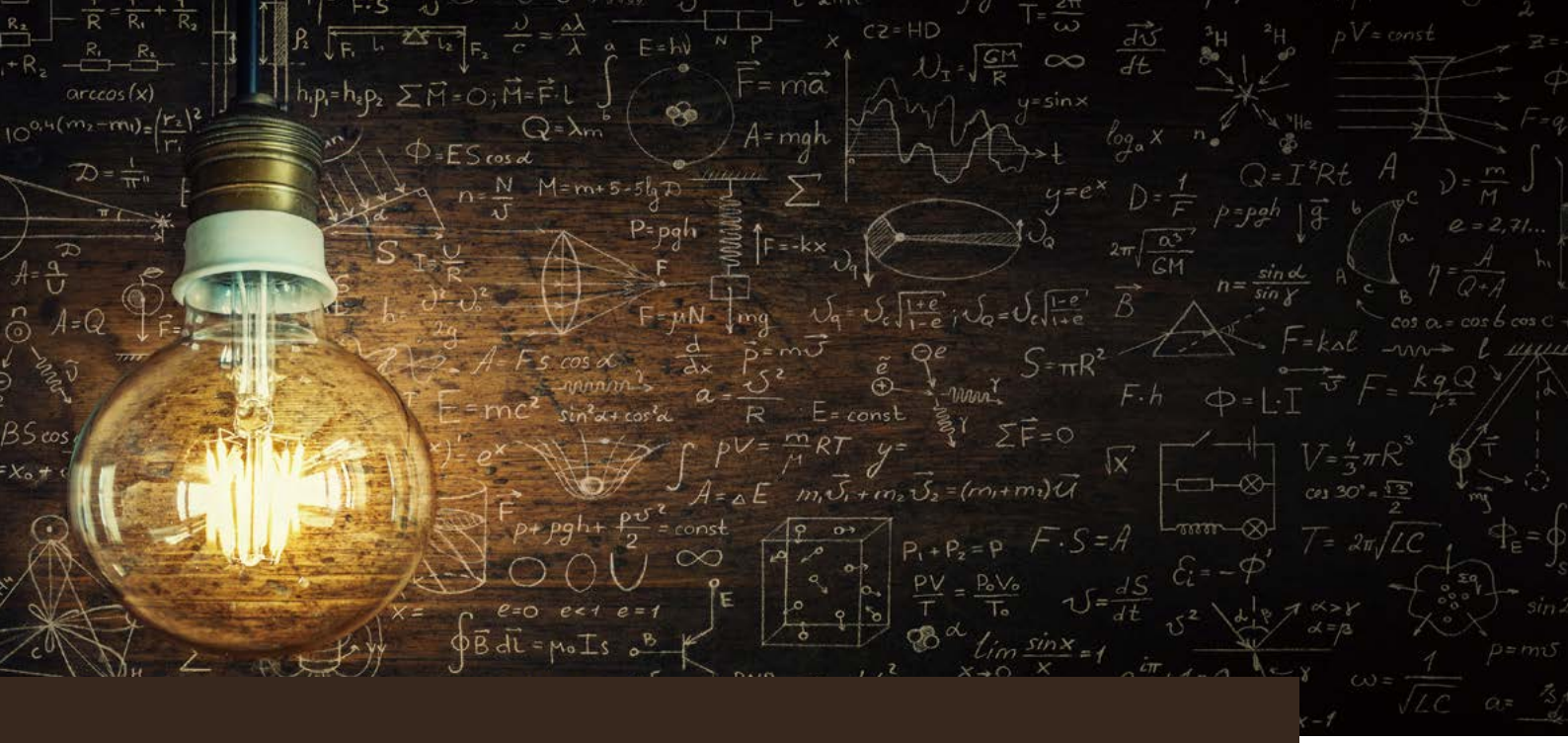
[8] Marius Iosifescu. *Finite Markov processes and their applications*. Courier Corporation, 2014.

[9] Christos H Papadimitriou. "On selecting a satisfying truth assignment". In: *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*. IEEE Computer Society. 1991, pp. 163–169.

[10] Ramamohan Paturi et al. "An improved exponential-time algorithm for *k*-SAT". In: *J. ACM* 52.3 (2005), pp. 337–364.

[11] T Schoning. "A probabilistic algorithm for k-SAT and constraint satisfaction problems". In: *Proceedings of FOCS 1999*. 1999, pp. 410–414.

[12] Yuren Zhou, Jun He, and Qing Nie. "A comparative runtime analysis of heuristic algorithms for satisfiability problems". In: *Artificial intelligence* 173.2 (2009), pp. 240–257.



Domain Theory Meets Interaction

Glynn Winskel
Edinburgh Research Center

Abstract

Domain theory and denotational semantics were formulated in the late 1960s by Dana Scott and Christopher Strachey, who together provided the first mathematical foundation for the semantics of programming languages and their analysis tools. Domain theory has been extremely influential and is still an active area of research, especially in its extensions to probabilistic, quantum programming, and differentiable programming. This article traces domain theory from its origins as a theory of computable functions, through its limitations, to its recent connections with interactive computation via concurrent games and strategies.

1 Introduction

After more than 50 years of development, domain theory permeates computer science. While it has limitations, domain theory has set a compelling paradigm in the formalization and analysis of computation. It interprets types as domains and programs as (continuous) functions. In many contexts, its models and reasoning techniques are the simplest we could hope for. If for this reason alone, domain theory is here to stay. It developed alongside the methodology of denotational semantics, which gives meaning to programming languages and systems in a compositional fashion. This methodology is the only way to manage the complexity of programming languages and systems, and its achievement tests our understanding and the robustness of our models.

The historical importance of domain theory is sometimes forgotten in the rush of practice and innovation. Domain theory and denotational semantics have played key roles in many areas. For example, they were critical in the evolution of functional programming languages, and they were where continuations (widely used in compilation) originated. Furthermore, abstract interpretation — a cousin that shares the same techniques — is at the foundation of many static analyzers, and linear logic, through domain theory, provides the important theory behind the popular system-programming language Rust. As this article explains, the techniques of domain theory and denotational semantics marry well with interactive computation and are still highly relevant today.

Domain theory grew out of a functional way of understanding computation. It began to meet its limits, unsurprisingly, with the shift to a more interactive form of computation. While the functional paradigm has a long history, theories of interactive computation have been more unsettled.

The article concludes with an indication of recent work that combines the methodology of domain theory with interaction through distributed/concurrent games. The role of domains is replaced by games and that of functions by strategies, so types are games and programs are strategies.

Central to any compositional theory of interaction is the dichotomy between a system and its environment. Concurrent games and strategies address the dichotomy in fine detail, in a local fashion, through distinctions between Player moves (events of the system) and Opponent moves (those of the environment).

A functional approach has to handle the dichotomy more ingeniously, through its blunter distinction between input and output (with the role of the environment captured as input). This has led to a variety of functional approaches that specialize in meeting particular interactive demands. Examples of such approaches include Girard's Geometry of Interaction; Gödel's Dialectica interpretation; lenses and optics; and the latter's extensions to containers in dependent lenses and optics. Surprisingly, at least for the author, domain-theoretic expressions of these functional approaches arise from special, rather simple, cases of concurrent games.

For a deeper dive into domain theory and its history, the author wrote the book "The formal semantics of programming languages", which is available in both English and Chinese [1]. And for a more advanced examination of domain theory, see the book chapter [2]. The latter part of this article, on concurrent games and strategies and their relation with functional paradigms, is expanded on informally in a newsletter [3] and more technically in [4].

I have tried to be as informal as possible in writing this article. In this article, additional information included to supplement understanding is marked in italics.

2 Early Beginnings

The history of domain theory is fairly well known. In the mid 1960s, Christopher Strachey realized he needed new techniques to understand the sophisticated programming languages he was developing — he needed a mathematical model with which to give semantics of programming languages. How else could he verify the correctness of his programs?

Over lunch, in Wolfson College, Oxford, the physicist Roger Penrose suggested that Strachey investigate the lambda calculus, a tool of logicians for describing and reasoning about computable functions. Meanwhile, the logician Dana Scott, at Princeton, was highly critical of the untyped nature of the lambda calculus. Specifically, he criticized it because it allowed such paradoxical phenomena as self-application (i.e., the application of a function to itself), which is forbidden in traditional set theory. The paradoxical combinator allowing recursive definitions in the lambda calculus relies on self-application. When Strachey and Scott met, there were bound to be fireworks of one kind or another. They got on very well, marking the beginning of their famous collaboration.

Scott persuaded Strachey to move to the safe typed lambda calculus and formalized a logic of computable functions (LCF) as a foundation for Strachey's ambitions. As a logician, Scott was concerned with mathematical foundations from the beginning. He introduced the idea that types denoted domains — simple forms of topological spaces — built on an order of approximation. Although a computable function can act on an infinite input (e.g., a computable function can act on a function on the natural numbers), it can only do so via finite approximations to the input. Accordingly, Scott promoted the idea that a computable function be understood as a continuous function from the domain of its input to the domain of its output. On domains, the usual topological definition of continuity amounted to the function preserving the approximation order and least upper bounds of chains.

The simplest form of a domain is a complete partial order, that is, a partial order $(D, \sqsubseteq_D)$ *of approximation with a least element* $\perp_D$ *and least upper bounds* $\bigsqcup_n d_n$*, a form of limit point, of chains of* $d_0 \sqsubseteq_D d_1 \sqsubseteq_D \cdots \sqsubseteq_D d_n \sqsubseteq_D \cdots$ *in* D*. A function* F *from a domain* $(D, \sqsubseteq_D)$ *to a domain* $(E, \sqsubseteq_E)$ *is continuous if* $F(d) \sqsubseteq_E f(d')$ *when* $d \sqsubseteq_D d'$*, and* $\bigsqcup_n F(d_n) = F(\bigsqcup_n d_n)$ *for any chain* $d_0 \sqsubseteq_D d_1 \sqsubseteq_D \cdots \sqsubseteq_D d_n \sqsubseteq_D \cdots$ *in* D*. Based on earlier ideas of Kleene* [1]*, a continuous function* F *from a domain to itself has a least fixed point* $\mathrm{fix}(\mathrm{F})$ *constructed as the least upper bound,* $\bigsqcup_n F^n(\perp) =: \mathrm{fix}(\mathrm{F})$*.*

One remarkable feature of domains and continuous functions is that, unlike topological spaces in general, under the pointwise order on functions the set of continuous functions from a domain D to a domain E itself formed a domain, the function space $[D \to E]$. Also, the product of domains $D \times E$, consisting of pairs of elements, formed a domain when ordered coordinatewise. In particular, a recursively defined function from D to E could be readily understood as a least fixed point of the continuous function F on $[D \to E]$ associated with the body of its recursive definition. LCF included a useful inequational logic for reasoning about recursive programs with tools such as Scott induction for establishing properties of least fixed points.

Scott's 1969 article on LCF was circulated widely and became very influential. However, it wasn't until much later, in 1993 [5], that it was published. This is because Scott had suddenly realized that the techniques for understanding recursive functions as least fixed points could be pushed to the level of types, thus providing a nontrivial domain D isomorphic to its domain of continuous functions $[D \to D]$ — a model of the (untyped) lambda calculus. His objections to the lambda calculus had vanished [6, 7]. Scott's discovery led to a flurry of activity.

3 The Word Spreads

Researchers outside Oxford joined in the effort. David Park, at Warwick, showed that in Scott's model, the paradoxical combinator of the lambda calculus denoted its least fixed point operator. A young Gordon Plotkin, at Edinburgh, and Scott independently discovered universal domains, within which all manner of types could be defined recursively by describing them as certain functions on the domain. Plotkin and Mike Smyth, then a postdoc at Warwick, extended Scott's ideas to a categorical treatment of recursively defined domains. This provided an understanding of a very broad range of recursive types. The mathematical foundations of functional programming were set.

Meanwhile, at Stanford, Robin Milner, Malcolm Newey and Richard Weyrauch had forged ahead with the mechanization of proofs in LCF [8, 9]. In the process, Milner invented the functional language ML to support assisted proofs securely. ML was inspired both by the functional nature of LCF itself and Peter Landin's language ISWIM [10].[1] With its watertight type discipline, ML ensured that only programs yielding legitimate LCF proofs would receive the type "Theorem". At Stanford, Milner began the push into the semantics of concurrent computation through "oracles" to settle nondeterministic choices. These roots continued at Stanford with the work of Zohar Manna and his student Jean Vuillemin on reasoning about recursively defined programs [11].

Domain theory forged new links between computer science and logic. Programming languages and computing systems were amenable to mathematical analysis. Computer scientists approached programming languages with a new confidence born out of a belief that sensible language constructs could be given a mathematical definition. Ultimately, the guidelines of domain theory influenced the design of programming languages and provided a foundation for functional programming.

By the mid 1970s, it seemed only a matter of time before domain theory could tackle all features of programming languages. Through continuations, Strachey and Christopher Wadsworth had shown how to provide semantics to jumps in imperative languages [12]. Plotkin, building on earlier ideas of Egli and Milner, had extended domain theory to a treatment of nondeterministic and parallel programs through his powerdomain [13] — his treatment avoided the non-associativity and non-commutativity of parallel

[1] ISWIM is Peter Landin's joke acronym for "If You See What I Mean."

composition Milner had met through using oracles. Nasser Saheb-Djahromi began constructing a probabilistic powerdomain, so enabling the denotational semantics of probabilistic programs. And young researchers in France were beginning to lend their own brilliant vision and mathematical expertise.

4 Limitations — Early Signs

Domain theory has given us a lasting vision of a mathematical approach to the semantics of programming languages. It supported the method of denotational semantics, whereby the semantics of a programming language is defined compositionally by structural induction on its syntax.

However, there were signs of domain theory's limits in very early work. Gilles Kahn showed how dataflow fitted easily within the scheme — it was a simple matter to represent dataflow processes as continuous functions from streams of input to streams of output, and handle loops in the network through the fixed point treatment of recursion that domain theory provided [14]. But, the extension to nondeterministic dataflow was to be problematic. The difficulties in giving a compositional semantics to nondeterministic dataflow was stressed much later in 1981 by Brock and Ackerman [15, 16]. While there are ways to give denotational semantics to nondeterministic dataflow, they lie outside traditional domain theory.

From LCF, Plotkin got the idea of studying its core programming language PCF (Programming Computable Functions), a tradition that continues in testing new theories with some extra feature or other in the presence of function spaces [17]. He invented the concept of **full abstraction** — the name is due to Milner. Full abstraction provides a way to formalize full agreement between the denotational and operational semantics of a language. Plotkin did this through the vehicle of PCF. Traditional domain theory did not provide a fully abstract model, because Scott's domains contained "parasitic" elements such as "parallel or," undefinable in PCF, on which operationally equivalent terms disagreed.

This sparked off the search for more operationally tuned domain theory, one that could capture the sequential evaluation of PCF and lambda-calculi. Both Milner and Vuillemin gave early, albeit slightly different, definitions of what it meant for a continuous function $f : D_1 \times \cdots \times D_m \to E_1 \times \cdots \times E_n$ between products of domains to be **sequential**: for a particular input $(x_1, \cdots, x_m)$, where $f(x_1, \cdots, x_m) = (y_1, \cdots, y_n)$, any increase in an output place y_j had to depend uniformly on an increase in a critical input place x_i. A problem with such a definition is that it depends on the particular way one decomposes a domain into a product.

This was remedied in 1975 by Kahn and Plotkin's definition of **sequential function** between **concrete domains**, in which there is an inbuilt notion of place (which they called a cell) [18]. Put simply, the elements of a concrete domain consist of a set of events, where an event is the filling of a cell with a particular value. As events occur, further cells become accessible and can be filled by at most one of several values. According to Kahn and Plotkin's definition, a sequential function between concrete domains is a continuous function for which, at any input, the filling of an accessible output cell depends on the filling of a critical accessible input cell. Note that there can be several critical cells. The problem was that space of sequential functions wasn't itself a concrete domain. Concrete domains didn't appear suitable for giving a denotational semantics to PCF.

Gérard Berry suggested two remedies. In his first, he proposed restricting continuous functions between domains to those that were **stable** — an approximation to being sequential [19]. Technically, stable functions preserve greatest lower bounds of compatible subsets of elements. More intuitively though, in a stable function, any part of the output depends on a unique minimum part of the input (reading "part of" as below in the order). Significantly, once the domains are axiomatized appropriately (producing what Berry called **dI-domains**), they have function spaces: the set of all stable functions between dI-domains, when ordered by a refinement of Scott's order (the **stable order**), forms a dI-domain. Two functions are in the stable order if they are in the Scott order and they share the same minimum inputs when producing common output. Berry went on to consider **bidomains**, which possessed both a Scott and stable order. Berry's stable domain theory received extra impetus in the mid 1980s with Jean-Yves Girard's use first of **qualitative domains** in models of polymorphism [20], and then again with his path-breaking discovery of linear logic through special kinds of dI-domains called **coherence spaces** related by stable functions [21].

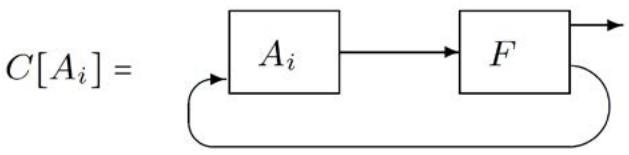
Stable functions have their own importance; however, as a treatment of sequentiality, they are just an approximation. With student Pierre-Louis Curien, Berry showed that there was a way to construct a domain theory for sequentiality, within which one could give a denotational semantics to PCF, albeit at the cost of departing from functions. They showed that concrete domains did indeed have a form of

function space if, instead of sequential functions, they used **sequential algorithms** [22]. Sequential algorithms can be expressed in terms of local decisions, deciding whether to output a value or inspect a cell for its value. Berry and Curien's pioneering work is a precursor to game semantics. As described by François Lamarche, a sequential algorithm is a form of strategy in which decisions of the sequential algorithm correspond to moves [23]. However, as Berry and Curien showed, a sequential algorithm can also be viewed as a sequential function together with an auxiliary function; given input and a cell accessible from the output, the auxiliary function returns a specific critical cell accessible at the input. (This characterization anticipated lenses in functional programming.)

In sequential algorithms, we begin to see a more interactive view of computation, not simply as the calculation of a function from input to output, but one in which the algorithm actively queries and makes demands on the input, and assigns values to cells. Without it being so obvious at the time, both sequential algorithms and stable functions were part of a growing chorus suggesting computation be based on interaction.

*The Brock-Ackerman anomaly of nondeterministic dataflow: As Kahn demonstrated, deterministic dataflow is a shining application of simple domain theory. However, nondeterministic dataflow is beyond its scope. This was an early indicator of the need for a more interactive view of computation. The compositional semantics of nondeterministic dataflow needs a form of generalized relation that specifies the **ways** input-output pairs are realized. This was shown in the early work of Brock and Ackerman, who were the first to emphasize the difficulties in giving a compositional semantics to nondeterministic dataflow, though our example is based on simplifications in the later work of Rabinovich and Trakhtenbrot [24], and Russell [25].*

Consider the following dataflow context:



There are two simple nondeterministic processes: A_1 and A_2 . While they both have the same input-output relation, they behave differently in the common feedback context $C[-]$, illustrated earlier. The context consists of a fork process F (which copies every input to two outputs), through which the output of the automata A_i is fed back to the input channel, as shown in the figure. Process A_1 behaves nondeterministically: Either it outputs a token and stops, or it outputs a token, waits for a token on input, and then outputs another token. Process A_2 has a similar nondeterministic behavior: Either it outputs a token and stops, or it waits for an input token and then outputs two tokens. For both processes, the input-output relation associates empty input to the eventual output of one token, and associates non-empty input to one or two output tokens. $C[A_1]$ can output two tokens, whereas $C[A_2]$ can output only one token. Notice that A_1 has two ways to realize the output of a single token from empty input, whereas A_2 has only one. It is this extra way, not caught in a simple input-output relation, that gives A_1 the richer behaviour in the feedback context.

*Over the years, there have been many solutions to giving a compositional semantics to nondeterministic dataflow. They all hinge on a form of generalized relation to distinguish the different ways in which output is produced from input. A compositional semantics can be given using **stable spans**, which extends Berry's stable functions to include nondeterminism [26]. Stable spans have re-emerged as a special form of concurrent strategy — see Section 8.*

5 Interaction

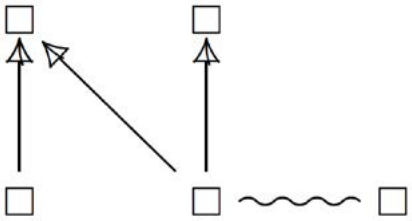
Concurrent processes can proceed independently but with points of interaction. Their treatment has long been a bugbear of traditional domain theory. While special cases such as deterministic dataflow could be expressed easily within domains, and Plotkin's powerdomains with a recursively defined domain of **resumptions** supported parallel composition through the nondeterministic interleaving of actions [13], in general the denotational semantics of parallel programs could seem convoluted. Indeed, after initial excursions into domain models, these complications led Robin Milner to forsake domain theory and denotational semantics in favor of a calculus of communicating systems (CCS) based on Plotkin's structural operational semantics and the process equivalence of bisimulation [27]. Instead, Tony Hoare, with Steve Brookes and Bill Roscoe, proposed a purpose-built domain of failure sets for communicating sequential processes (CSP) [28]. Both Hoare and Milner settled on synchronization, possibly with the exchange of values, as their primitive of communication. Concurrency became a relatively separate field of study for a number of years, and it remains somewhat syntax-driven.

Since the early 1960s, Carl Adam Petri and others worked on developing a radically new model of computation, **Petri nets**. Petri nets are based on events making local changes to conditions representing local states. A state of a Petri net is captured by a **marking**, which picks out those conditions that currently hold. The Petri net's dynamics — how one marking changes to another — is based on the key idea that the occurrence of an event ends the holding of its preconditions (those conditions with arrows leading to the event) and begins the holding of its postconditions (those conditions with arrows from the event).

The structures defined by Petri were remarkably similar to those Kahn and Plotkin had uncovered as representations of concrete domains in their investigations of sequentiality. In fact, through the intermediary concept of **event structure**, comprising a set of event occurrences with relations of causal dependency and conflict, Mogens Nielsen, Gordon Plotkin, and the author of this article were able to transfer concepts across the two communities, around Petri nets and domains. Just as transition systems unfold to trees, so Petri nets unfold to event structures [29, 30]. Notably, Petri's notion of confusion freeness in Petri nets coincided with the restrictions Kahn and Plotkin were making to localize nondeterministic choice to cells. A little later, it was realized that Berry's dI-domains were the domains of event structure configurations ordered by inclusion [31, 32].

*In its simplest form, an **event structure** is $(E, \leq, \#)$, comprising a partial order of **causal dependency** $\leq$ and a symmetric, irreflexive binary relation of **conflict** # on **events** E. The relation $e' \leq e$ expresses that event e causally depends on the previous occurrence of event e'. $e\#e'$ means that the occurrence of one event, e or e', excludes the occurrence of the other. Together, the relations satisfy two axioms: An event causally depends on only a finite number of events, and events that causally depend on conflicting events are themselves in conflict. A state or history of an event structure is caught in the definition of configuration: **configuration** of an event structure consists of a subset of an event which is down-closed with regards to $\leq$ and conflict-free. Two events are considered to be causally independent — referred to as **concurrent** — if they are not in conflict and neither one causally depends on the other.*

Diagrammatically, events are depicted by squares, immediate causal dependencies by arrows, and immediate conflicts by wavy lines. For example,



represents an event structure with five events. The event on the far right is in immediate conflict with one event (as shown by the wavy line). It is also in conflict with all events except that on the lower far left, with which it is concurrent.

Whereas Petri nets were largely used to model concurrent processes, the corresponding structures in domain theory were being used as representations of domains (types of processes). This was a curious mismatch. The reconciliation of these two views came much later in generalizations of domain theory, in which both types and processes denoted event structures — as is the case in concurrent games and strategies.

From the burgeoning world of richly structured models and their equivalences in concurrency, it became clear that concurrent computation wasn't going to fit neatly within traditional domain theory. For instance, a Petri net carried much more structure than could be supported by a point in a domain of information.

Fortunately, category theory helped organize models for concurrency: an individual model (e.g., Petri nets, event structures, or a transition system) carried its own style of map to form a category. The maps represented a form of event-respecting simulation and could be used, for instance, to relate the behavior of a parallel composition to that of its components. Relations between the different categories of models could be expressed as adjunctions, and they helped systematize the equivalences on concurrent processes [33, 34]. This separated models of concurrency from the syntax and operational semantics in which they were so often embedded.

For example, a map of event structures from E to E' is a partial function f on events which respects configurations and events: it sends a configuration x of E, by direct image, to a configuration fx of E' such that no two distinct events in x go to the same event in fx. While causal dependency need not be preserved by f, it is reflected locally: if the $e, e' \in x$ and $f(e) \leq f(e')$, then $e \leq e'$. Consequently, maps of event structures automatically preserve the concurrency relation on events.

This taxonomy was based on existing models. However, it suggested a more general class of models with the

versatility to be adapted in the same way as domain theory — a form of generalized domain theory [35, 36]. In several early domain models of processes, a process had been identified with the set of computation paths it could perform. One well-known model of this kind is Hoare's "trace" model of CSP, in which a process denotes the set of sequences of visible actions it can perform. The generalized domain theory was similar, but instead of a process being a **set** of computation paths, it represented a process by a **presheaf** of computation paths. Essentially, a presheaf is like a generalized characteristic function but where the usual truth values are replaced by sets, to be thought of as sets of ways to realize truth. By modelling a process as a presheaf, one allowed for the process possibly following several computation paths of the same shape and kept track of how the paths of the process branched nondeterministically. Presheaf models for concurrency connected concurrent computation with rich mathematics, in particular the mathematics of species. However, their operational reading could be challenging. Sometimes, a denotational semantics in terms of presheaves could be represented by event structures. Technically, the category of elements of the presheaf denotation took the form of the configurations of an event structure. By bringing the role of event structures to the fore, this eventually led to a game semantics based on event structures — see Section 8.

6 French Influence

Jean-Yves Girard is an imposing figure in French logic and computation. He has a distrust of what he sees as the too simplistic and over-arching use of algebra to structure and analyze logic. He has been been highly critical of Alfred Tarski's definition of truth in a model for first-order logic, and indeed about denotational semantics, to which his work has nevertheless contributed enormously.[1] Girard's work emphasizes an operational understanding of proof and computation. This is far from saying it forsakes mathematical models, or concentrates on syntax in the way of traditional proof theory. On the contrary, the models he has developed and inspired have considerable ingenuity and depth, and they have shifted interest to new ways of understanding proof and computation.

Through his reinvention of stable domain theory in the more restricted setting of coherence spaces, Girard was led to the important discovery of **linear logic** in the mid 1980s [21]. This gave a deconstruction of traditional logic into a more fundamental resource-conscious logic. That work helped turn the emphasis of domain theory away from function spaces and "currying", with respect to a product, to function spaces with respect to more general tensor products. In the jargon of category theory, it shifted the emphasis from cartesian-closed to monoidal-closed categories. In semantics of computation, we now see models of linear logic everywhere. Girard's coherence spaces correspond to a very special form of event structure in which causal dependency is the trivial identity relation.

In studying the proofs of linear logic, Girard discovered **geometry of interaction** (GoI) [38]. Although originally explained in terms of the mathematical structures of quantum mechanics, GoI was shown by Samson Abramsky and Radha Jagadeesan to have a more traditional, domain-theoretic reading in which the mechanism of interaction was achieved in the manner of Kahn networks, using least fixed points [39]. GoI was related to Jean-Jacques Lévy's optimal reduction of the lambda-calculus by Martin Abadi, George Gonthier, and Lévy and has influenced the implementation of programming languages, notably via token-based computation [40]. Today, GoI is perhaps most often viewed as an early form of game semantics — see Section 8.3.

7 Game Semantics

There was some vagueness about what was entailed in realizing a solution to the full-abstraction problem for PCF.

The 1980s had seen several technical successes through the use of representations of domains: concrete data structures [18] and event structures [32] to give an operational description of how functions compute; and information systems, owing to Scott and the author of this article, to give a logical presentation and considerably simplify the recursive definition of domains and their logical relations [41–43]. Girard's work also often exploited the fine structure of such representations.

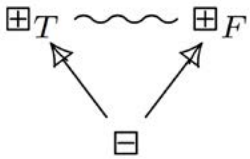
Given this history, it is not surprising that several early attempts to construct a fully abstract model for PCF were based on adjoining extra structure to domains or their representations. Some examples are the use of both the Scott order and stable order on functions in bidomains and bistructures [44], via Ehrhard's hypercoherences [45], and O'Hearn and Riecke's powerful logical relations [46]. These attempts were put paid to, or at least compromised, by Ralph Loader. In a tour de force, Loader showed that the

[1] For instance, in domain models of polymorphism [37] and through linear logic [21].

full-abstraction problem for PCF, as originally understood, couldn't be achieved effectively. In other words, the presentation of a fully abstract domain model for PCF would be non-computable [47].

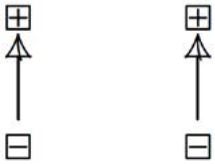
This left open the intermediate question of whether there were other more independently motivated models in which all the finite elements were definable by PCF terms. From this, a (non-effective) domain model could then be obtained by quotienting. To this question, called "intensional full-abstraction", two different affirmative answers were given and pioneered the highly informative use of games in the semantics of programming languages. Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria invented AJM games [48], while Martin Hyland, Luke Ong, and independently Hanno Nickau discovered HO games [49]. In many ways, game semantics fitted the bill for a more operationally tuned domain theory — the role of domains was replaced by games, and that of continuous functions was replaced by strategies. The role of games was extended beyond functional to imperative programs.

To give a flavor of the game semantics of programs, we present games and strategies in a slightly unconventional way as event structures. They are instances of concurrent games and strategies described in Section 8. We first describe the game generally associated with the type of Booleans as an event structure. In this game, the first move is by Opponent (representing the environment) asking for a value, the event ⊟. Player (representing the program) may then respond to this with a move providing a value, either True (represented by the event $\boxplus_T$) or False (represented by the conflicting event $\boxplus_F$):



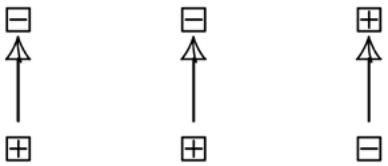
With an eye to illustrating type constructions on games, let's simplify the game. Imagine a type with a single value. We can represent this by a game in which Opponent's initial move is to ask for a value, the event ⊟. Player may then respond to this with a move providing a value, the event ⊞. We can draw this game as the event structure:

Suppose we want to represent the type of pairs of values as a game. We can do this simply by putting two copies of the single-value game in parallel, forming the event structure:

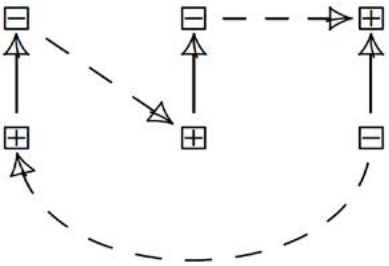


Opponent may ask for a value on the left, a value on the right, or both, to which Player may or may not respond.

*For the next step, we ask how we can represent the type of programs from pairs to a value, a form of function type, as a game. While it is typical to consider a strategy **in** a game, the idea of a strategy **from** one game **to** another is less well known but still quite widely used.[1] According to this idea, explained a little further in Section 8, a strategy from the game of pairs to the single-value game is a strategy in the following game:*



The game is built as the parallel composition of the two games (one for input of pairs and the other for output), but with the role of Player and Opponent reversed in the game for pairs. A strategy in this game is essentially a program. For example, consider the following event structure: it represents the strategy (or program) which first inspects input in the left component of a pair, then inspects the right component before yielding an output value.



In general, a strategy in the preceding game, if it responds to the initial move of Opponent, will either output a value directly or demand more input from which to play further. Hopefully this gives an idea of how programs can be viewed as strategies. As here, a great deal of traditional game semantics is not concerned with winning conditions, though they have a role in describing programming languages (e.g., in ensuring a program is correct or progresses).

[1] The author first came across this idea in a talk John Conway gave on Numbers and Games in the early 1970s to the Archimedeans — the mathematical society of the University of Cambridge.

However, the game story was far from complete. For one thing, it wasn't clear, at least initially, how to reconcile the two different versions, AJM and HO, of game semantics. For another, more significantly, the games were based on sequential plays in which Player and Opponent moves alternated.

The bias toward sequentiality has handicapped the theories of games in general. In **game theory**, this bias has led to a menagerie of different kinds of games, with each kind of game specialized to cope with one feature or another. In this case, concurrency is handled in a piecemeal fashion, often through extra structure to capture imperfect information, limiting the ways that the games and strategies of game theory can be composed. While game semantics is very much concerned with structure and composition, its games are predominantly sequential. In most cases, concurrency is represented indirectly via the interleaving of atomic actions of the participants. This rarely does justice to the distributed nature of the system described, inhibits analysis of its causal dependencies, and is often accompanied by compensatory, *ad hoc* fairness assumptions.

What was lacking was a rich algebraic theory of distributed/concurrent games in which Player and Opponent are more accurately thought of as teams of players, distributed over different locations and able to move and communicate. Although there are glimpses of such a theory in earlier work [50–54], a considerable unification occurs with the systematic use of event structures to formalize concurrent games and strategies through their causal structure [55, 56].

8 Concurrent Strategies

Distributed/Concurrent games answer the need to rethink the foundations of games, foundations that provide a more flexible grounding where such games truly belong, in the models and theories of interaction of computer science. The driving idea is to replace the role of game trees in more traditional developments with event structures.

A **concurrent game** is represented by an event structure together with a polarity marking its events to identify whether they are moves of Player, marked + (the system), or Opponent, marked − (the unknown environment). Games often have extra features such as winning conditions, describing those configurations at which Player wins, or a payoff functions assigning a reward to each configuration. For simplicity, here we assume that games are **race-free**. Specifically, there is no immediate conflict between a Player and an Opponent move. While there may be a conflict between Player and Opponent moves, such a conflict must be inherited either from the conflict between two Player moves or from that between two Opponent moves.

With games as event structures, the history of a gameplay is no longer described by a sequence of moves, but instead by a partial order expressing their causal dependency. The transition from total to partial order brings in its wake technical difficulties and potential for undue complexity unless it's done artfully. Fortunately, one can harness the mathematical tools developed for interacting processes, specifically on event structures [32, 33].

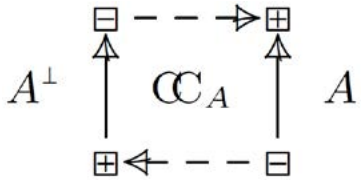
There are two fundamentally important operations on two-party games. One is that of forming the **dual** game, in which the roles of Player and Opponent are interchanged. On an event structure with polarity A, this amounts to reversing the polarities of events to produce the dual $A^{\perp}$. A strategy in a game implicitly refers to a strategy for Player. A strategy for Opponent, known as a counterstrategy, in a game A is identified with a strategy in $A^{\perp}$. The other operation is a **parallel composition** of games, achieved on event structures A and B by simply juxtaposing them, making events from different components non-conflicting, to form $A \| B$.

Following the ideas described by Conway and Joyal [57, 58], a strategy σ **from** a game A **to** a game B is taken to be a strategy **in** the compound game $A^{\perp} \| B$. Given another strategy τ from the game B to a game C, the **composition** $\tau \odot \sigma$ is provided essentially by playing the two strategies against each other over the common game B, and then hiding that interaction.

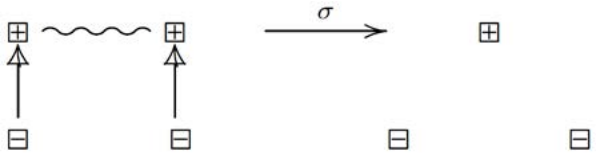
But what is a strategy in a concurrent game? The article [55] provides a detailed answer and its rationale. Essentially, a definition of strategy is chosen precisely to make the copycat strategy the identity with regards to composition. A strategy in a concurrent game prescribes moves for Player and their dependencies; it should both obey the constraints of the game and not constrain Opponent's behavior beyond the constraints of the game. The axioms on strategies entail a formal connection with presheaf models mentioned earlier in Section 5 and through them with Scott domains [59]. In general, strategies may be nondeterministic, but we can restrict the scope to deterministic strategies in which all nondeterministic behavior stems from Opponent.

As an example, consider the **copycat** strategy in the game $A^{\perp} \| A$, which — following the spirit of a copycat — has Player copy the corresponding Opponent moves in the other

component. The copycat strategy $\mathbb{CC}_A$ is obtained by adding extra causal dependencies to $A^\perp \| A$, so that any Player move in either component causally depends on its copy — an Opponent move — in the other component. This is illustrated below when A is a simple game comprising a Player move causally dependent on a single Opponent move:



Strategies are not always obtained by simply adding extra causal dependencies to the game. In general, a strategy in a game A is expressed as a map of event structures $\sigma : S \to A$ describing the choices of Player moves by the event structure S. For example, consider a game comprising two Opponent moves in parallel with one Player move, and the strategy for Player is to make its move if Opponent makes a move. The example is represented by the following map



This takes the two conflicting Player moves on the left to the single Player move on the right, and takes each Opponent move on the left to the corresponding Opponent move on the right.

*Not all maps of event structures $\sigma : S \to A$ are strategies. There are two further axioms on maps if they are to be deemed strategies: **receptivity** and **(linear) innocence**. Intuitively, they prevent Player from constraining Opponent's behavior further than is allowed by the game. Receptivity expresses that any Opponent move allowed from a reachable position of the game is present as a possible move in the strategy. Innocence says a strategy can only adjoin new causal dependencies of the form $\boxminus \rightarrowtriangle \boxplus$, where Player awaits moves of Opponent beyond those inherited from the game. Silvain Rideau and the author of this article have shown that the axioms are precisely those that make copycat the identity for the composition of strategies [55].*

*A strategy **from** a game A **to** a game B is a strategy **in** the compound game $A^\perp \| B$. As such, a strategy is a map $\sigma : S \to A^\perp \| B$. Given another strategy $\tau : T \to B^\perp \| C$ from B to a game C, the **composition** $\tau \odot \sigma$ is obtained by playing the two strategies off against each other over the game B. To do this precisely, it is useful to harness two operations associated with maps of event structures: (1) **pullback**, to produce the **interaction** $\tau \circledast \sigma : T \circledast S \to A^\perp \| B \| C$, which "synchronizes" matching moves of S and T over the game B; and (2) a **partial-total factorization** of partial maps of event structures, to hide the synchronizations and produce, as its defined part, the strategy **composition** $\tau \odot \sigma$, namely, $T \odot S \to A^\perp \| C$.*

*A strategy $\sigma : S \to A$ is **deterministic** if all conflict in S is inherited through causal dependency on conflicting Opponent moves. Deterministic strategies can be composed. That copycat is deterministic, so an identity for the composition of deterministic strategies, is due precisely to games being **race-free**. The strategy illustrated earlier is not deterministic.*

8.1 Winning Conditions

The winning conditions of a game A take the form of a subset W of its configurations. An outcome in W is a win for Player. A strategy for Player is **winning** if it always prescribes moves for Player to end up in a winning configuration, regardless of Opponent's activity or inactivity [60].

*Formally, a strategy $\sigma : S \to A$ is **winning** if σx is in W for all +-maximal configurations x of S. A configuration is +-maximal if no additional Player moves can occur from it. This can be shown to be equivalent to all plays of σ against counterstrategies of Opponent resulting in a win for Player.*

As the dual of a game A with winning conditions W, we again reverse the roles of Player and Opponent to get $A^\perp$ and take the winning conditions to be the set-complement of W. In a simple parallel composition of games with winning conditions $A \| B$, Player wins if they win in either component. With these extensions, we can take a winning strategy from a game A to a game B — where both games have winning conditions — to be a winning strategy in the game $A^\perp \| B$. The choices ensure that the composition of winning strategies is winning. Because games are race-free, copycat will always be a winning strategy.

8.2 Imperfect Information

In a game of **imperfect information**, some moves are masked (or inaccessible), and strategies with dependencies on unseen moves are ruled out. One can extend games with imperfect information in a way that respects the operations of concurrent games and strategies [61]. Each move of a game is assigned a level in a global order of access levels. Moves of the game or its strategies can only causally depend on moves at equal or lower levels.

In more detail, a fixed preorder of ***levels*** $(\Lambda, \leq)$ *is presupposed. A* Λ***-game*** *comprises a game* A *with a* ***level function*** $l : A \to \Lambda$ *such that, if* $a \leq_A a'$*, then* $l(a) \leq l(a')$ *for all moves* a, a' *in* A*. A* Λ***-strategy*** *in the* Λ***-game*** *is a strategy* $\sigma : S \to A$ *for which if* $s \leq_S s'$ *then* $l\sigma(s) \leq l\sigma(s')$ *for all* s, s' *in* S*. The access levels of moves in a game are left undisturbed in forming the dual and parallel composition of games. As before, a* Λ*-strategy from a* Λ*-game* A *to a* Λ*-game* B *is a* Λ*-strategy in the game* $A^{\perp} \| B$*. It can be shown that* Λ*-strategies compose together to give a* Λ*-strategy.*

8.3 Old from New

The additional complexity of event structures over trees shouldn't obscure direct connections between strategies on concurrent games and the more familiar notions on games as trees. Event structures subsume trees. An event structure is **tree-like** when any two events are either in conflict or one causally depends on the other. In this case, its configurations form a tree with regards to inclusion, with root the empty configuration.

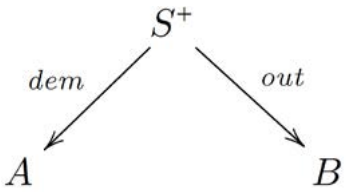
A **tree-like** game is one for which its underlying event structure is tree-like. Because games are race-free, at any finite configuration of a tree-like game, the next moves (if any) are purely those of either Player or Opponent. In this sense, positions of a tree-like game belong to either Player or Opponent. At each position belonging to Player, a deterministic strategy chooses either a unique move or to stay put. In contrast to many presentations of games, in a concurrent strategy, Player isn't forced to make a move (however, that can be encouraged through suitable winning conditions). Winning conditions specify those configurations at which Player wins. Consequently, in a tree-like game, such conditions can be both finite and infinite branches in the tree of configurations.

Clearly then, the dual of a tree-like game is tree-like. A counterstrategy, as a strategy in the dual game, picks moves for Opponent at its configurations. When the counterstrategy is deterministic at each Opponent configuration, it chooses to stay or make one particular move. As expected, the interaction determines a finite or infinite branch in the tree of configurations, which in the presence of winning conditions will be designated as a win for one of the two players.

On tree-like games, we recover familiar notions. More surprising is that, by exploiting the richer structure of concurrent games, we can recover other familiar paradigms, not traditionally tied to games, or if so only somewhat informally.

For example, by restricting to deterministic strategies between concurrent games where all moves are Player moves, we rediscover Berry's stable domain theory, of which Girard's qualitative domains and coherence spaces are special cases. For such restricted games, general, possibly nondeterministic strategies correspond to **stable spans**, a model discovered and rediscovered in compositional accounts of nondeterministic dataflow.

In more detail, consider a strategy σ *from one such purely Player game* A *to another* B*. This is a map* $\sigma : S \to A^{\perp} \| B$ *which is receptive and innocent. Notice that in* $A^{\perp} \| B$ *all the Opponent moves are in* $A^{\perp}$ *and all the Player moves are in* B*. By receptivity any configuration of* A *is copied as possible initial input in* S*. By innocence, the only new immediate causal connections, beyond those in* $A^{\perp}$ *and* B*, that can be introduced in* S *are those from Opponent moves of* $A^{\perp}$ *to a Player move in* B*. Beyond the causal dependencies inherited from the two games,* S *can only make a Player move in* B *causally depend on a finite subset of moves in* $A^{\perp}$*. For this reason any Player move* s *in* S *is associated with both an output event* $out(s)$ *in* B *and a demand on input* $dem(s)$*, a finite configuration of* A *needed for* s *to occur. A strategy between purely Player games corresponds to a* ***stable span*** *involving two (special) functions from the event structure* S^{+} *obtained by restricting* S *to Player moves. For a general strategy, the output events for a particular input may be in conflict—the output is nondeterministic. When* σ *is deterministic, all conflicts are inherited from conflicts between Opponent moves. Then the strategy* σ *corresponds to a* ***stable function*** *from the domain of configurations of* A *to the domain of configurations of* B*.*



Only marginally more complicated than those purely Player games are games that consist of two parallel components, one a purely Player game and the other with purely Opponent moves. The deterministic strategies between such games yield the Abramsky-Jagadeesan model for GoI built on stable functions [39].

Adjoining winning conditions and imperfect information to these games, so Opponent can see the moves of Player but not the converse, we recover a dialectica category [62]

(i.e., Gödel's dialectica interpretation) from deterministic strategies. Dialectica categories, studied by Valeria de Paiva in her Cambridge PhD, mark an early occurrence of "lenses" used in functional programming, where they were invented independently to make composable local changes on data-structures [64, 65].

Generalizing dialectica games a little, they become examples of "container types", and deterministic strategies become "dependent lenses" [66]. Lenses generalize to "optics" in the context of general tensor products [67]. Optics based on stable spans are recovered when we allow the strategies to be nondeterministic.

Because concurrent games and strategies have been enriched, for example, to provide semantics to probabilistic and quantum programs [68–71], such enrichments remain for these special cases. For example, they show how to add probability to strategies between dialectica games. All of this is dealt with in more detail in [4], and informally and more accessibly in [3].

The preceding examples concern ways of handling interaction within a functional approach. They have evolved, often independently. Any compositional theory of interaction is forced to handle the dichotomy between a system and its environment. Concurrent games and strategies address the dichotomy in fine detail, through polarities on events. As the examples show, a functional approach has to handle the dichotomy much more ingeniously, through its cruder distinction between input and output — with basic interaction treated through the application of a function to its argument. Within concurrent games, we can more clearly see what separates and connects the differing paradigms.

The examples do not cover all the ways in which functions are extended to cope with interaction. A notable omission in this article is the theory of effects in programming languages. That theory uses the technology of monads and algebraic theories to refine the influential work of Eugenio Moggi and, essentially, to describe computation in terms of enriched computation trees [72, 73]. Concurrent strategies have been enriched to address probabilistic and quantum computation. There is further work to do in connecting concurrent games and strategies with the theory of effects.

9 Conclusion

Although this article discusses the limitations of domain theory, it also demonstrates the theory's lasting power. Concurrent games and strategies provide a general model of interaction. Their generality can provide guidance in the form a model or its enrichment should take. In special cases, they simplify to easier domain models. In one direction, concurrent strategies help build domain models. In the other, when domain models are available, they can simplify concurrent strategies enormously. It would be foolish to use a complicated model when a simpler one is available. In many contexts, domain theory provides the simplest models we know!

Acknowledgements

My thanks to the anonymous reviewers for their helpful suggestions.

References

[1] Winskel, G.: The formal semantics of programming languages, an introduction. MIT Press. In Italian (UTET Libreria, 1999). In Chinese (China Machine Press, CITIC Publishing House, 2004). (1993)

[2] Abramsky, S., Jung, A.: Domain theory. Handbook of logic in computer science **3** (1994) 1–168

[3] Winskel, G.: Domain theory and interaction. Newsletter BCS-FACS FACTS Issue 2021-2 July (compressed) (2021)

[4] Winskel, G.: Making concurrency functional. CoRR **abs/2202.13910** (2022)

[5] Scott, D.S.: A type-theoretical alternative to ISWIM, CUCH, OWHY. Theor. Comput. Sci. **121**(1&2) (1993) 411–440

[6] Scott, D.S.: Mathematical concepts in programming language semantics. In: American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1972 Spring Joint Computer Conference, Atlantic City, NJ, USA, May 16-18, 1972. Volume 40 of AFIPS Conference Proceedings., AFIPS (1972) 225–234

[7] Scott, D.S.: Data types as lattices. SIAM J. Comput. 5(3) (1976) 522–587

[8] Milner, R.: Implementation and applications of scott's logic for computable functions. In: Proceedings of ACM Conference on Proving Assertions About Programs, Las Cruces, New Mexico, USA, January 6-7, 1972, ACM (1972) 1–6

[9] Gordon, M.J.C., Milner, R., Morris, L., Newey, M.C., Wadsworth, C.P.: A metalanguage for interactive proof in LCF. In Aho, A.V., Zilles, S.N., Szymanski, T.G., eds.: Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages, Tucson, Arizona, USA, January 1978, ACM Press (1978) 119–130

[10] Landin, P.J.: The next 700 programming languages. Commun. ACM 9(3) (1966) 157–166

[11] Manna, Z., Vuillemin, J.: Fix point approach to the theory of computation. Commun. ACM 15(7) (1972) 528–536

[12] Strachey, C.S., Wadsworth, C.P.: Continuations: A mathematical semantics for handling full jumps. High. Order Symb. Comput. 13(1/2) (2000) 135–152

[13] Plotkin, G.D.: A powerdomain construction. SIAM J. Comput. **5**(3) (1976) 452–487

[14] Kahn, G.: The semantics of a simple language for parallel programming. In Rosenfeld, J.L., ed.: Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5-10, 1974, North-Holland (1974) 471–475

[15] Brock, J.D., Ackerman, W.B.: Scenarios: A model of non-determinate computation. In Díaz, J., Ramos, I., eds.: Formalization of Programming Concepts, International Colloquium, Peniscola, Spain, April 19-25, 1981, Proceedings. Volume 107 of Lecture Notes in Computer Science., Springer (1981) 252–259

[16] Winskel, G.: Events, causality and symmetry. Comput. J. **54**(1) (2011) 42–57

[17] Plotkin, G.D.: LCF considered as a programming language. Theor. Comput. Sci. **5**(3) (1977) 223–255

[18] Kahn, G., Plotkin, G.D.: Concrete domains. Theor. Comput. Sci. **121**(1&2) (1993) 187–277

[19] Berry, G.: Stable models of typed lambda-calculi. In: ICALP. Volume 62 of Lecture Notes in Computer Science., Springer (1978) 72–89

[20] Girard, J.: The system F of variable types, fifteen years later. Theor. Comput. Sci. **45**(2) (1986) 159–192

[21] Girard, J.: Linear logic. Theor. Comput. Sci. **50** (1987) 1–102

[22] Berry, G., Curien, P.: Sequential algorithms on concrete data structures. Theor. Comput. Sci. **20** (1982) 265–321

[23] Curien, P.: On the symmetry of sequentiality. In Brookes, S.D., Main, M.G., Melton, A., Mislove, M.W., Schmidt, D.A., eds.: Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7-10, 1993, Proceedings. Volume 802 of Lecture Notes in Computer Science., Springer (1993) 29–71

[24] Rabinovich, A.M., Trakhtenbrot, B.A.: Communication among relations (extended abstract). In Paterson,

M., ed.: Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings. Volume 443 of Lecture Notes in Computer Science., Springer (1990) 294–307

[25] Russell, J.: Full abstraction for nondeterministic dataflow networks. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, Los Alamitos, CA, USA, IEEE Computer Society (nov 1989) 170–175

[26] Saunders-Evans, L., Winskel, G.: Event structure spans for nondeterministic dataflow. Electr. Notes Theor. Comput. Sci. 175(3): 109-129 (2007)

[27] Milner, R.: A Calculus of Communicating Systems. Volume 92 of Lecture Notes in Computer Science. Springer (1980)

[28] Brookes, S., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. J. ACM **31** (1984) 560–599

[29] Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures and domains. TCS **13** (1981) 85–108

[30] Winskel, G.: Events in computation. (1980) PhD thesis, University of Edinburgh.

[31] Winskel, G.: Event structure semantics for CCS and related languages. In: ICALP'82. Volume 140 of LNCS., Springer, A full version is available from Winskel's Cambridge homepage (1982)

[32] Winskel, G.: Event structures. In: Advances in Petri Nets. Volume 255 of LNCS., Springer (1986) 325–392

[33] Winskel, G., Nielsen, M.: Models for Concurrency. In: Handbook of Logic in Computer Science 4. OUP (1995) 1–148

[34] Joyal, A., Nielsen, M., Winskel, G.: Bisimulation from open maps. Inf. Comput. **127**(2) (1996) 164–185

[35] Hyland, M.: Some reasons for generalising domain theory. Mathematical Structures in Computer Science **20**(2) (2010) 239–265

[36] Cattani, G.L., Winskel, G.: Profunctors, open maps and bisimulation. Mathematical Structures in Computer Science **15**(3) (2005) 553–614

[37] Girard, J.: The system F of variable types, fifteen years later. Theor. Comput. Sci. 45(2) (1986) 159–192

[38] Girard, J.: Towards a geometry of interaction. Contemporary Mathematics **92** (1989) 69–108

[39] Abramsky, S., Jagadeesan, R.: New foundations for the geometry of interaction. Inf. Comput. **111**(1) (1994) 53–119

[40] Gonthier, G., Abadi, M., Lévy, J.J.: The geometry of optimal lambda reduction. In: POPL '92. (1992)

[41] Scott, D.S.: Domains for denotational semantics. In Nielsen, M., Schmidt, E.M., eds.: Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings. Volume 140 of Lecture Notes in Computer Science., Springer (1982) 577–613

[42] Winskel, G., Larsen, K.G.: Using information systems to solve recursive domain equations effectively. In Kahn, G., MacQueen, D.B., Plotkin, G.D., eds.: Semantics of Data Types, International Symposium, Sophia-Antipolis, France, June 27-29, 1984, Proceedings. Volume 173 of Lecture Notes in Computer Science., Springer (1984) 109–129

[43] Abramsky, S.: Domain theory in logical form. In: Proceedings of the Symposium on Logic in Computer Science (LICS '87), Ithaca, New York, USA, June 22-25, 1987, IEEE Computer Society (1987) 47–53

[44] Plotkin, G.D., Winskel, G.: Bistructures, bidomains and linear logic. In Abiteboul, S., Shamir, E., eds.: Automata, Languages and Programming, 21st International Colloquium, ICALP94, Jerusalem, Israel, July 11-14, 1994, Proceedings. Volume 820 of Lecture Notes in Computer Science., Springer (1994) 352–363

[45] Ehrhard, T.: Hypercoherences: A strongly stable model of linear logic. Math. Struct. Comput. Sci. **3**(4) (1993) 365–385

[46] O'Hearn, P.W., Riecke, J.G.: Kripke logical relations and PCF. Inf. Comput. **120**(1) (1995) 107–116

[47] Loader, R.: Finitary PCF is not decidable. Theor. Comput. Sci. **266**(1-2) (2001) 341–364

[48] Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Inf. Comput. 163(2): 409-470 (2000)

[49] Hyland, J.M.E., Ong, C.H.L.: On full abstraction for PCF: I, II, and III. Inf. Comput. 163(2): 285-408 (2000)

[50] Abramsky, S., Melliès, P.A.: Concurrent games and full completeness. In: LICS '99, IEEE Computer Society (1999)

[51] Laird, J.: Game semantics for higher-order concurrency. In Arun-Kumar, S., Garg, N., eds.: FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings. Volume 4337 of Lecture Notes in Computer Science., Springer (2006) 417–428

[52] Melliès, P.A., Mimram, S.: Asynchronous games : innocence without alternation. In: CONCUR '07. Volume 4703 of LNCS., Springer (2007)

[53] Ghica, D.R., Murawski, A.S.: Angelic semantics of fine-grained concurrency. In: FOSSACS'04, LNCS 2987, Springer (2004)

[54] Faggian, C., Piccolo, M.: Partial orders, event structures and linear strategies. In: TLCA '09. Volume 5608 of LNCS., Springer (2009)

[55] Rideau, S., Winskel, G.: Concurrent strategies. In: LICS 2011. (2011)

[56] Winskel, G.: ECSYM Notes: ECSYM notes: event structures, stable families and concurrent games. http://www.cl.cam.ac.uk/~gw104/ecsym-notes.pdf (2016)

[57] Conway, J.: On numbers and games. Wellesley, MA: A K Peters (2000)

[58] Joyal, A.: Remarques sur la théorie des jeux à deux personnes. Gazette des sciences mathématiques du Québec, 1(4) (1997)

[59] Winskel, G.: Strategies as profunctors. In: FOSSACS 2013. Lecture Notes in Computer Science, Springer (2013)

[60] Clairambault, P., Gutierrez, J., Winskel, G.: The winning ways of concurrent games. In: LICS 2012: 235-244. (2012)

[61] Winskel, G.: Winning, losing and drawing in concurrent games with perfect or imperfect information. In: Festschrift for Dexter Kozen. Volume 7230 of LNCS., Springer (2012)

[62] de Paiva, V.: The Dialectica categories. PhD Thesis, University of Cambridge (1988)

[63] Avigad, J., Feferman, S.: Gödel's functional ("dialectica") interpretation. (1999) 337–405

[64] Oles, F.J.: A category theoretic approach to the semantics of programming languages. PhD Thesis, University of Syracuse (1982)

[65] Foster, J.N., Greenwald, M.B., Moore, J.T., Pierce, B.C., Schmitt, A.: Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. ACM Trans. Program. Lang. Syst. **29**(3) (2007) 17

[66] Abbott, M.G., Altenkirch, T., Ghani, N.: Containers: Constructing strictly positive types. Theor. Comput. Sci. **342**(1) (2005) 3–27

[67] Pickering, M., Gibbons, J., Wu, N.: Profunctor optics: Modular data accessors. Art Sci. Eng. Program. **1**(2) (2017) 7

[68] Winskel, G.: Distributed probabilistic and quantum strategies. Electr. Notes Theor. Comput. Sci. 298: 403-425 (2013)

[69] Paquet, H., Winskel, G.: Continuous probability distributions in concurrent games. In Staton, S., ed.: Proceedings of the Thirty-Fourth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2018, Dalhousie University, Halifax, Canada, June 6-9, 2018. Volume 341 of Electronic Notes in Theoretical Computer Science., Elsevier (2018) 321–344

[70] Clairambault, P., de Visme, M., Winskel, G.: Game semantics for quantum programming. Proc. ACM Program. Lang. **3**(POPL) (2019) 32:1–32:29

[71] Clairambault, P., de Visme, M.: Full abstraction for the quantum lambdacalculus. Proc. ACM Program. Lang. **4**(POPL) (2020) 63:1–63:28

[72] Moggi, E.: Computational lambda-calculus and monads. In: Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989, IEEE Computer Society (1989) 14–23

[73] Plotkin, G.D., Power, A.J.: Computational effects and operations: An overview. Electron. Notes Theor. Comput. Sci. **73** (2004) 149–163

Formulation of Open Source Strategies and Measurement of Community Building

Peixin Hou, Zi Li, Yehui Wang
Open Source Management Center

Abstract

With the development of the digital industry, more and more enterprises are embracing open source and incorporating it into their development strategies. Centering on the formulation of open source strategies and the building of enterprise-class open source communities, this paper explains how enterprises can leverage open source to develop their ecosystem economy and promote digital transformation. In addition, this paper proposes community measurement methods for community developers and managers to help enterprises build high-growth and high-loyalty open source communities in support of their open source strategies.

Keywords

open source strategy, open source community, community measurement

1 Introduction

With early forms of software, source code was widely disclosed and spread across academic and scientific research institutions. Even suppliers and customers often used source code for upstream and downstream delivery. However, at that time, this could not be termed "open source software", because it lacked basic elements such as open source licenses, open source communities, and collaborative contributions.

Before the term "open source software" appeared, free software had become prevalent and developed into a movement. At this time, basic principles (four basic elements of freedom) came into being, and relevant organizations, groups, and star projects — such as the Free Software Foundation (FSF) and GNU (GNU's Not Unix!) — also emerged. Later, in order to gain more support from business organizations and further explore practical value in the business world, the term "open source software" was formally proposed [1]. This paper will not dive into the origins or political standing of open source software and free software [2]. Today, most people use the term "open source software" to refer to both concepts on most occasions.

Open source software has become ubiquitous. The industry has deepened its understanding of open source software, and it continues to gain more and more knowledge. The major participants in open source have gradually shifted from individuals to enterprises, ultimately leading to a truly closed-loop business model (Project -> Product -> Profit -> Project) and promoting the sustainable development of open source. Nowadays, enterprises embrace open source to not only reduce costs, but also quickly develop new products — this remains a key driver for many enterprises (Figure 1). In recent years, with the development of the ecosystem economy, digital transformation, and "software-defined everything", more and more enterprises want to achieve the aforementioned goals more quickly through open source (Figure 2).

Centering on the formulation of open source strategies and the building of enterprise-class open source communities, this paper explains how enterprises can leverage open source to develop their ecosystem economy and promote digital transformation. In addition, this paper proposes community measurement methods for community developers and managers to help enterprises build high-growth and high-loyalty open source communities in support of their open source strategies.

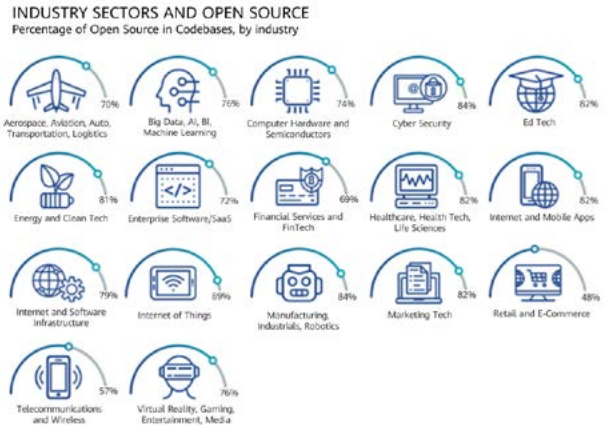


Figure 1 Proportion of open source uses in each industry [source: OSSRA]

Figure 2 Proportion of proactive open source in vertical industries

2 Formulation of Open Source Strategies

Generally, the following six elements need to be considered in the formulation of an open source strategy [3]: business model, intellectual property rights (IPR) strategy, open source technology strategy, project governance and community platform strategy, ecosystem building strategy, and success measurement. This section focuses on the first five elements. Section 3 "Measurement of Community Building" describes the last element.

2.1 Element 1: Business Model

What business models to support and how to monetize them are the fundamental basis for supporting other elements and must therefore be determined at the outset. Enterprises do not open source out of charity. Without business monetization, open source will not obtain continuous investment, not to mention development.

Typically, open source supports the following business models:

- Subscription and support (SnS): Sell subscription services

centered on open source code and provide commercial support. Red Hat Enterprise Linux (RHEL) and Pivotal Cloud Foundry adopt this model.

- Feature enhancement and charging (open core): Open only the base version, but charge for additional components of proprietary code. Prominent examples include GitLab Enterprise and products from numerous device vendors, including Huawei GaussDB and commercial software built on OpenStack, K8S, or other open source software.
- Associated products (traffic diversion): Utilize open source software to reach more users and divert traffic to other services. For example, Android Open Source Project (AOSP) and Mozilla Firefox provide traffic diversion for Google Search.
- Dual software licenses: Provide software with both a free and a proprietary software license. A free license, which is also a copyleft open source license, is used to attract users and co-builders. An example is the GNU General Public License (GPL). With a proprietary software license, although it involves charges, users are not restricted by open source licenses like GNU GPL and can enjoy SLA assurance and services from commercial companies. Linux desktop system Qt is a good example.
- (Cloud) services: Sell open source software as cloud services, such as Amazon OpenSearch Service and Google Kubernetes Engine.
- Hardware: Sell hardware products but open related software on the hardware to attract and create convenience for developers. For example, Intel Clear Linux OS supports and enables Intel x86 CPUs.

It is worth noting that an open source project is not a commercial product, even if they share the same source code. When building a business model for a commercial product based on an open source project, consider the following aspects:

- Compared with the open source project, the product must have sufficient business value.
- Compared with the open source project, the product must be positioned more clearly.
- The product needs to compete with software in the open source project.
- Different companies may compete using software from the same open source projects.

To explain these aspects, we will use RHEL as an example. Because RHEL always adheres to the ''Upstream First'' or even ''Upstream Only'' principle, all of its source code comes from upstream communities. It even incorporates self-developed features into upstream communities prior to its own products. To RHEL, the following points are of particular importance:

First, unique business value. Linux's server operating system (OS) is complex. Through SnS, RHEL provides enterprise-class services for customers in terms of usage, management, configuration, and maintenance, as well as various northbound/southbound authentication mechanisms. Open source projects cannot provide such business value.

Second, clear product positioning. RHEL achieves a trade-off between advanced community features and OS stability as well as long-term maintenance and support, which is important for enterprises. Compared with upstream open source projects that focus on new feature development and limited version maintenance in a short period, RHEL has clearer product positioning and it better meets the requirements of enterprise users.

Third, compete with software from other projects. Although RHEL has unique value and positioning compared with upstream projects, it is inevitable that some users may directly obtain software from upstream projects and integrate, install, and maintain the software themselves. Instead of ignoring this, RHEL built (or rather, acquired) the CentOS community with RHEL versions but removed all RHEL trademarks from it. As RHEL versions are periodically maintained and updated in the CentOS community, users can enjoy the same product quality and maintenance as with RHEL free of charge. Although CentOS appears to be competing with RHEL, it actually attracts and retains potential RHEL customers, and helps prevent a significant number of users from leaking away to self-built OSs or competitors.

Fourth, compete with other products based on the same open source project. In the field of commercial Linux distributions, RHEL faces competition from SUSE Linux Enterprise, Ubuntu, and others. These products depend on the same open source project and therefore have similar features. After long-term competition and cooperation, they gradually form their own customer groups and ecosystems. Although RHEL dominates the market, these products will co-exist for a long time.

2.2 Element 2: IPR Strategy

Open source software has complete IPR. In this sense, intellectual achievements in the public domain are not open source software, because they do not have related IPR.

Formulating IPR for open source software involves professional legal issues. Therefore, this paper does not provide any detailed guidance on how to formulate IPR strategies. Instead, we discuss four aspects that you should consider with your IPR lawyers or legal professionals when developing IPR strategies:

- Open source license

  A license must be selected for open source software that is proactively opened to the public. (To adapt to different scenarios or integrate with different open source software, multiple licenses may be selected for the same open source software.) In general, once the software creator selects an open source license, the code contributed by other contributors will also use the selected license. Integrated distributions are an exception. Take openEuler as an example. As an OS distribution, openEuler integrates thousands of pieces of open source software from native upstream communities. Because the licenses of these pieces of software may be different, the compatibility between the licenses determines whether different software can be integrated. Given that developers may introduce new open source software to enhance OS functions in the future, the license compliance strategy will be reviewed on a per-order basis and updated periodically. Generally, integrated distribution communities adopt the good/bad licensing mode. For example, the Fedora and openEuler communities both have similar mechanisms (https://fedoraproject.org/wiki/Licensing:Main#Good_Licenses).

  Open source licenses also have other challenges, including the identification of incomplete licenses in software, diversity of license application granularities (software, module, or file level), diversity of licensed scenarios (even the same license has different rights and obligations in different scenarios such as user mode or kernel mode and static link or dynamic link), and changes to software licenses. Such challenges require the in-depth involvement of relevant tool systems (https://compliance.openeuler.org/) and legal professionals.

- Copyright

  Regardless of whether open source software is hosted to an open source foundation for open governance or controlled by the software company itself, its copyright ownership belongs to the original developer or employer of the original developer and will not be transferred with open source contribution. Open source grants a wide range of rights — such as using, copying, distributing, and modifying software — to users (including open source foundations) under the restrictions of licenses or similar contributor agreements (CLA or DCO). Sometimes, these rights are even permanent, non-exclusive, and non-reclaimable. (For details, see the content of open source licenses and OSI's definition of open source software.) However, granting of such rights does not mean that the copyright is transferred. Misunderstandings often arise about this, especially when the original developer or enterprise donates the software to an open source foundation for open governance. This is also common practice in mainstream open source foundations, including the Linux Foundation, Apache Foundation, and OpenStack Foundation. Nevertheless, certain open source foundations, such as FSF (https://www.gnu.org/licenses/why-assign.en.html), do require copyright transfer. This is largely due to its ''software freedom'' principle and the convenience of rights protection. Such a copyright strategy is not proactively adopted across the industry (especially large enterprises). It should be noted that, even in FSF, the transfer of copyright is not mandatory. Rather, it is performed only when contributors want FSF to take full charge of copyright protection. Recently, we noticed that some FSF-owned projects had canceled this strategy; for example, GCC no longer requires transferring copyright (https://news.slashdot.org/story/21/06/05/0246247/gcc-will-no-longer-require-copyrights-be-assigned-to-the-fsf).

- Patent right
  - Some licenses clearly authorize free use of patents, but we still need to be prudent that open source may eliminate our right to sue for patents.
  - For key patent fields, Huawei should be cautious about contributing and participating in open source software.

- In terms of patent isolation, the most thorough method is to set up separate companies (like Qualcomm, Ericsson, and Microsoft did) that are focused on open source contributions.

- Trademark
  - If an open source project is donated to an open source foundation, the ownership of the project trademark will also be transferred.
  - If the project trademark conflicts with a product trademark, prepare a new project trademark before contributing it to the foundation.
  - The right to use a project trademark, especially in open governance scenarios (hosted to foundations), is generally common to all users. No exclusive rights can be reserved.

2.3 Element 3: Open Source Technology Strategy

The open source technology strategy is divided into two dimensions:

- Technical selling points

  There are countless open source projects. The key to standing apart from others lies in how competitive and advanced our technology is.

  It must be able to solve real-world problems:

  - From unavailable to available (O3DE): Before Open 3D Engine (O3DE) was introduced, mainstream 3D game engine software available commercially was closed-source. While there were a very limited number of open source engines available in the market, they either were 2D engines or could not provide AAA-level image quality for gaming. After acquiring CryEngine code, AWS refactored, enhanced, and opened the source code to break the industry landscape and quickly migrate O3DE computing to cloud.
  - From available to better (AOSP): Before AOSP appeared, Linux-based mobile OSs had several pain points, including: 1) Key components lacked high-quality open source software. For example, the telephone subsystem RIL, browser, and Java virtual machines needed to be purchased from commercial companies. 2) The versions and models of open source components were so fragmented that they could not form unified app development interfaces. As a result, upper-layer app development could not be normalized, and no unified app market was available for app distribution and monetization. 3) There was a lack of interconnection with emerging cloud services, such as map and search services. 4) There was a lack of man-machine interfaces (MMIs) that represented next-generation capacitive touchscreen sliding experience (iPhone had such MMIs). In some cases, third-party components needed to be developed or purchased. The launch of AOSP solved these problems. This explains why AOSP attracts so many northbound/southbound developers and its ecosystem grows so rapidly.
  - Great vision (ECOMP): Figure 3 illustrates the vision of telecom network cloudification.

Figure 3 Vision of telecom network cloudification

- What to open and what to close

  Take RHEL as an example again. Based on the "Upstream First" and "Upstream Only" principles, all RHEL code comes from upstream communities. How can RHEL build differentiated competitiveness?

  For this, RHEL does not open the following:

  - Building a tool chain, including compilation options (to gain competitiveness 1: binary optimization of commercial RHEL versions).
  - Developing test suites and related tools (to gain competitiveness 2: efficiency of vulnerability fix in commercial RHEL versions).

2.4 Element 4: Project Governance and Community Platform Strategy

- Project governance: including open governance and closed governance. In Figure 4, the horizontal axis represents whether the participants are individuals

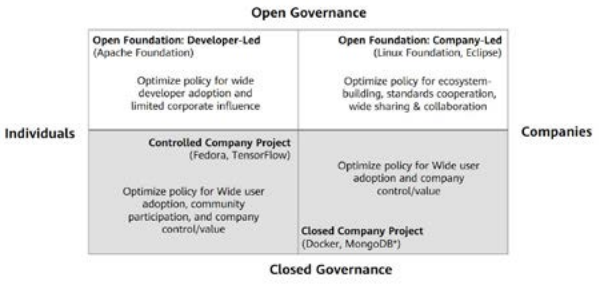


Figure 4 Open governance model

or companies, and whether the community accepts developers from other companies. There is no relationship between whether the community is controlled by its own company [3].

- Code engineering and community operation platform: including code hosting platforms, software engineering tools, and operation tools (DevOps pipeline, CICD, contribution check & access control, developer portal, and official website), as shown in Figure 5.

Some tools may have service continuity risks. The following policies might help mitigate such risks:

- Gradually switch commercial tools/services to trusted suppliers.
- Convert open source components to commercial services in China.
- Gradually shift to independent maintenance and evolution of risky open source components.

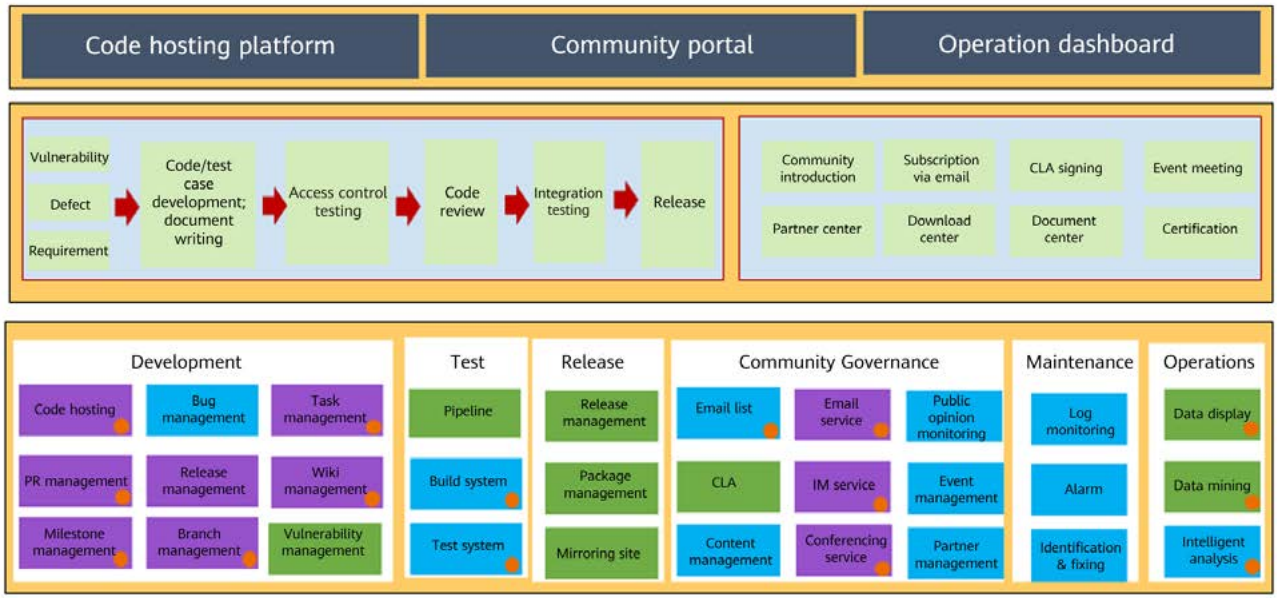


Figure 5 Code engineering and community operation platform

2.5 Element 5: Ecosystem Building Strategy

Community leaders need to guide different types of ecosystem partners to discover the value of participating in their communities. The following describes the strategies for four common types of ecosystem partners [3]:

- User:
  - Develop the user ecosystem by increasing adoption.
  - Create a complete distribution for user-centric projects.
  - Focus on promoting downloads and integrating with other open source ecosystems.
- Developer:
  - Generally, it is best to contribute open source projects to open source foundations to attract more developers with open governance.
  - Focus on frameworks, platforms, or tools rather than complete distributions, helping improve developer productivity.
  - Focus on easy integration with other open source development.
- Other companies (including competitors):
  - Select an existing foundation to reduce the legal work brought by newcomers.
  - "Project development first, community adaptation next". Here are a few examples: Kubernetes -> Cloud Native Computing Foundation (CNCF), OpenStack -> OpenStack Foundation, Ceph -> Ceph Foundation, and Apache Incubator -> Apache Project.
  - Kick off projects through prominent channels. For example, Kubernetes was kicked off at DockerCon 2014, and OpenStack was kicked off at Open Source Convention (OSCON) 2010.
  - Make intentions known in advance to recruit members. For example, the Linux Foundation announced its intention to set up Hyperledger in 2015, with the setup starting the next year.
  - Recruit end users with vendors to ensure that the project provides value and it is promoted. For example, the Hyperledger project was jointly launched with CME Group, Deutsche Börse, State Street, and Wells Fargo.
  - Communicate with other companies in advance to agree who will lead and who will maintain the architecture. (Figure 6 uses the Cisco OpenDayLight project as an example.)

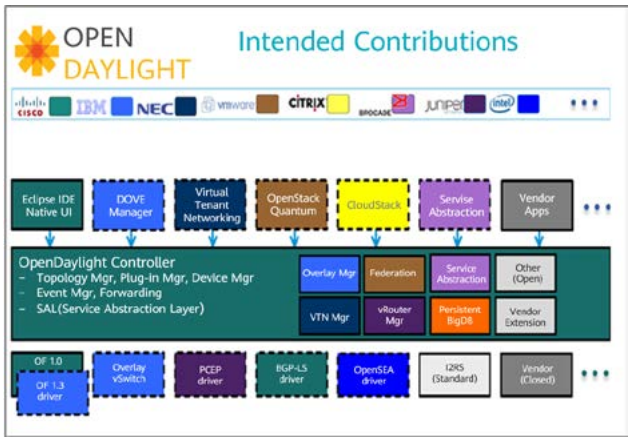


Figure 6 Contributors in the OpenDayLight project

- Standards and industry organizations
  - Reference implementation in open source software: EFSP (JCP) -> Jakarta EE
  - Leading standards: Docker -> OCI

2.6 Element 6: Success Measurement

Success can be measured from two dimensions: final business results and community development process. The business result dimension is relatively simple and is an ultimate measurement for closed-loop open source strategy formulation and execution effect. However, it takes a long time to obtain business results, making it difficult to master the process and implement prompt adjustments. In terms of the second dimension, this section provides several intermediate result measurements and their potential impact on business decision-making.

- User quantity measurement (website portal or user survey):
  - Number of users/downloads
  - Community size
  - Number of users/companies (Company X has downloaded our project 200 times, so we can try to sell the product to it.)
  - Users' geographical locations (30% of downloads are made by Chinese users, so we should open a sales office in China.)

- Measurement of other ecosystem activities (foundation measurement platform or community insight tool):
  - Members
  - Activities
  - Contributors
  - Training
  - Developer activeness
  - Downstream projects

3 Measurement of Community Building

3.1 Two Measurement Perspectives

''Community'' refers to a group of people or organizations who gather for the common good. Fundamentally, creating a successful community means creating an ecosystem where members can do meaningful work and quickly improve themselves. Their enduring motivation for growth is the key to building a constantly popular community.

After the open source strategy of the project is determined and announced to the public, we need to build a community. The ultimate goal is to run a sustainable open source community with high growth and loyalty. During community building and operations, a host of problems may occur. Such problems may include no reply to issues for a long time, poor developer satisfaction, and developer churn. To solve these problems, a complete community measurement system must be built.

The purpose of community measurement is to determine whether the community fulfills all set goals in different phases. Specifically, we need to develop an effective quantitative solution, design expectations based on measurement standards, and improve the solution based on timely insights into measurement results. This ensures that the community will develop in a healthy direction.

The success of a community can be measured from two perspectives: developer and community manager.

3.2 Measurement from a Developer's Perspective

Members are the core force of a community, and developers are the most important role among members. Therefore, efficient developers are one of the key factors for a community to succeed.

Before explaining how to conduct measurement from a developer's perspective, we briefly describe the basic principles for creating member portraits [6]:

- Determine the portrait types of key members based on the vision, mission, and phased goals of the community. Such types include user, evangelist, event organizer, issue supporter, and developer portraits. Note that the roles of some portraits may overlap. Because portrait creation is time-consuming work, pay more attention to the quality rather than quantity of portrait types.
- When creating member portraits, focus on the following elements: capabilities, experience, motivation, concerns, expected rewards, and fields of interest.

The following describes how to create a developer portrait.

3.2.1 Developer Portrait

Developer portrait creation is a practice of user portrait technology in community development settings. Developer portraits consist of various features of community developers and their relationships.

- Developer features are usually a series of labels. In Figure 7, each label represents a skill of the developer.

Figure 7 Developer's skill cloud

Contribution, as another feature of developers, refers to what a developer has done for a specific community. It reflects how active and proficient the developer is in the community. Contribution covers all activities performed by the developer, such as participating in projects or committing code on the Gitee platform, or answering questions in the Stack Overflow community.

In Figure 8, each label represents a project in which the developer has participated in the community. The diameter of each circle indicates the contribution of the developer to the project. The larger the diameter, the more they have contributed.

Figure 8 All the projects that a developer has contributed to

- Their association with other developers can be represented by a developer relationship graph, which visualizes the interaction between developers to quantify the collaboration strength within the community. Using the developer relationship graph, we can identify the trend of developers' — especially core developers' — participation in a community, and determine whether developers tend to leak away from the community. As shown in Figure 9, the developer relationship graph is generally divided into social relationship, direct collaboration relationship, and indirect collaboration relationship [4].

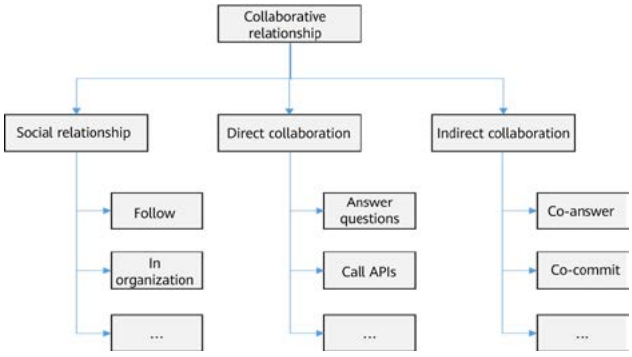


Figure 9 Developer relationship graph

Each type of relationship encompasses detailed collaboration relationships. The relationship type and relationship strength are viewed as attributes of collaboration relationships. In the development activities of developers, social relationship refers to the interpersonal relationships between developers in a specific community. Although not directly related to development activities, a relevant study found that joining the same projects as friends did would significantly increase developers' contribution rate [5]. Social relationships can be classified into ''Follow'' and ''In organization'' relationships, indicating whether developers are following each other and whether they belong to the same organization, respectively. Direct collaboration means that developers directly interact with each other. That is, when handling the same task, two developers need to closely communicate and cooperate with each other. Assuming that developers D1 and D2 have ''Answer to'' and ''Call API'' relationships, D1 must have answered questions raised by D2, and when implementing functions, D1 called some interfaces provided by D2. The frequency of such collaboration is expressed numerically. Such close collaboration relationships may also be embodied in many activities (e.g., modifying the same code files, and developing and testing the same code). Indirect collaboration means that developers indirectly interact with each other. It is a weak collaboration relationship compared with direct collaboration, but it plays a role in the overall development task. For example, if developers D1 and D2 both answer a question or commit code to an open source project, they are in the ''Co-answer'' and ''Co-commit'' relationships.

3.2.2 Community Participation Journey Framework

When developer portraits are finished, we need to establish a community participation journey framework (Figure 10) to measure the value created by developers in different phases [6].

The framework covers three key parts of community participation. Different types of member portraits require different community participation journey frameworks.

Figure 10 Participation journey framework for a developer community

- Guidance for newcomers: The left of the pentagram represents what the community can do to help newcomers create value in the simplest way. The value belongs to both newcomers and the community.
- Status transition: On the right of the pentagram, the developer starts to switch between three key identities, i.e., casual, regular, and core.
- Incitement: To promote community development and maintain member participation, we will take a series of measures (blue boxes in the figure) to help community members accumulate experience, develop new skills, and maintain activeness.

Following the creation of developer portraits and community participation journey frameworks, the next step is to introduce a community portrait maturity model (Figure 11). This model provides a set of tools to define the success of developer portraits in each phase.

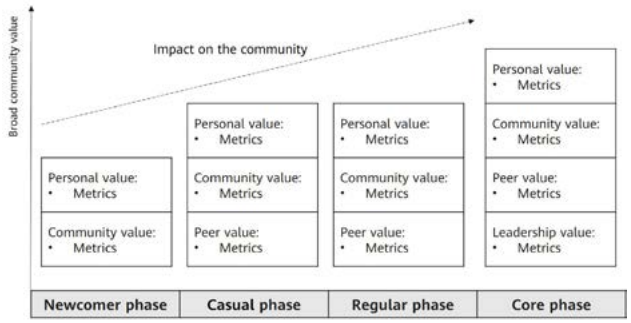


Figure 11 Community portrait maturity model

Along the horizontal axis are the phases in community participation journey frameworks, including newcomer, casual, regular, and core. Success criteria need to be defined for each phase, and metrics in different dimensions are required to facilitate measurement.

- Personal value: What measurable value should a member portrait bring to the member? For example, for the newcomer phase, the success criteria for the issue supporter portrait can be defined as "propose new issues in the community and obtain answers."
- Community value: What measurable value does a member portrait bring to the community? For example, for the newcomer phase, the success criteria for the issue supporter portrait can be defined as "solve the issues proposed by other community members."
- Peer value: What measurable value does a member portrait bring to other members in the community? For example, for the regular phase, the success criteria can be "provide one-on-one guidance or training for other members in the community."
- Leadership value: What measurable value does a member portrait bring to community leadership?

3.2.3 Developer NPS Measurement Model

To measure the value that developers create for the community in different phases (newcomer/casual/regular/core), a net promoter score (NPS) measurement model needs to be built from the perspective of developers [7]. Through measurement and insights provided by this model, the community and developers can bring more value to each other. NPS is a common metric adopted by the industry to measure user experience. It is an effective tool for managing user experience and improving customer satisfaction. Ultimately, it asks the question: How likely is it that you would recommend our community to a friend or colleague, and why? The NPS approach clearly classifies users into three categories and calculates an easy-to-understand value (Figure 12) based on which you can make improvements and achieve success.

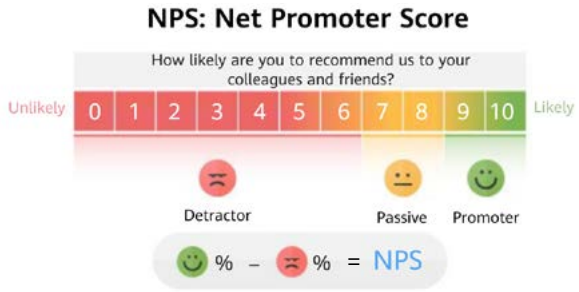


Figure 12 Definition of NPS

The developer NPS measurement model (Figure 13) consists of two parts: NPS measurement system and the NPS feedback system. The NPS measurement system, when

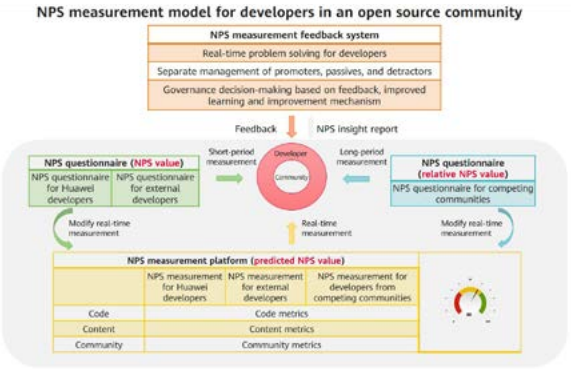


Figure 13 Developer NPS measurement model

used in conjunction with developer portraits, can identify problems in a community and developers' requirements through the NPS questionnaire. The NPS feedback system gives prompt solutions and adjusts the governance mechanism of the community. In this way, both developers' satisfaction and loyalty to the community are improved.

3.3 Measurement from a Community Manager's Perspective

From the perspective of community managers, many metrics and models can be used to measure the success of an open source community. The problem is that open source projects face an unparalleled amount of data — any objects with obtainable data can be collected and tracked. The metrics that each organization tracks — and how they deal with data — largely depend on the strategic goals of their community and the unique challenges in the marketplace and community. In particular, large organizations are unlikely to track all objects and derive meaning from it, since they have so many projects. To eliminate this dilemma, managers need to consider the strategic goals of their communities when selecting metrics and domains [8].

3.3.1 Metric Selection

Setting strategic goals for a community involves the following: design the vision and mission of the community, specify the value that the community provides for people/organizations, and set phased goals. During implementation, the work results of the community need to be measured.

When setting specific metrics, ensure that goals can be clearly and accurately measured — never be ambiguous. In this sense, we need to adhere to the following rules [6]:

- Measure key dimensions.

  Communities have seven key dimensions:

  - Growth rate: How many newcomers have joined the community? How does it change with time?
  - Retention rate: How many people continuously participate in community activities?
  - Activeness of community members
  - Activeness of collaboration between organization and community members
  - Delivery capability: Have all contributors in the community created value according to their responsibilities?
  - Attendance rate: How many people attend online and offline activities?
  - Effectiveness: Is the operation mechanism of the entire community smooth enough?

- Measure both "quality" and "quantity".

  Indeed, tracking the quantity of different types of contributions is a good way to measure activeness and delivery rate. The quality of contributions is also an important aspect. We need to measure both quality and quantity.

- Quantify measurement.

  When measuring whether goals are fulfilled, we should be able to answer with a clear "Yes" or "No" instead of "Maybe".

- Limit the number of metrics.

  To keep things simple and focused, reduce the number of metrics and track only key metrics.

3.3.2 Support from Core Organizations

As community founders and leaders, core organizations need a broad range of skills, covering aspects such as planning, development, interaction, and community experience improvement, if they want to truly create a successful community. In addition to these skills being developed, support systems also need to be established within the organizations.

The process for an organization to develop these skills can be divided into three phases [6]:

1. Incubate new skills: Create a resource, education, strategy, and execution environment that can introduce new skills, provide training for personnel, and help members apply these skills.
2. Establish mechanisms: After learning from the experience and lessons in the incubation phase, establish mechanisms, deepen understanding within teams, and improve standards.
3. Integrate: Promote these mechanisms to all teams in the organization.

By combining these three phases with the target skills of

professional domains, we propose the following organization capability maturity model.

Organization Capabilities

	1. Incubation	2. Mechanism	3. Integration
Strategy	Assist in planning and enforcing strategies	Regular structured planning	Strategy evolution and execution
Management	Clarify the community management style	Team development and coaching	Full support from the organization
Growth	Develop an initial growth strategy	Verify growth pattern assumptions	Identify growth patterns and further develop them
Participant	Community teams/employees	Other key departments	The entire organization
Leadership	Specific teams/employees	Stakeholders in the organization/community	Independent leadership team
Tool	Community tools are available	Community tools are integrated into broader organizational resources	Community tools become a key part of core product/engineering work
Metric	Metrics related to target member portraits	Regular improvement and execution	Community metrics become an important part of organizational KPIs

Figure 14 Organization capability maturity model

As shown in Figure 14, the organization capability maturity model is split across seven rows top down, each representing a professional domain. Each professional domain spans all three phases of skill development.

When using this model, organizations need to periodically review it, collect feedback from related departments, and integrate improvement plans into the next iteration.

3.4 Community Experience Improvement

Based on the relationships between community health measurement systems and developer portraits, we worked together with Professor Wei Wang from East China Normal University and Professor Jian Cao's team from Shanghai Jiao Tong University to improve the experience of the openEuler and MindSpore communities led by Huawei.

3.4.1 Developer Retention Through Portraits

In the openEuler community, by providing skill labels (Figure 15) for developers in different special interest groups (SIGs), we found that most developers had participated in CVE vulnerability fixing. With this knowledge, community managers can develop a simpler vulnerability fixing process and introduce automated vulnerability identification and fixing tools to greatly improve developer experience.

In the MindSpore community, we built a developer issue interaction graph (Figure 16), helping community managers identify more than 10 inactive developers. Then through questionnaires and offline communications, the managers

Figure 15 Developer labels in the openEuler community

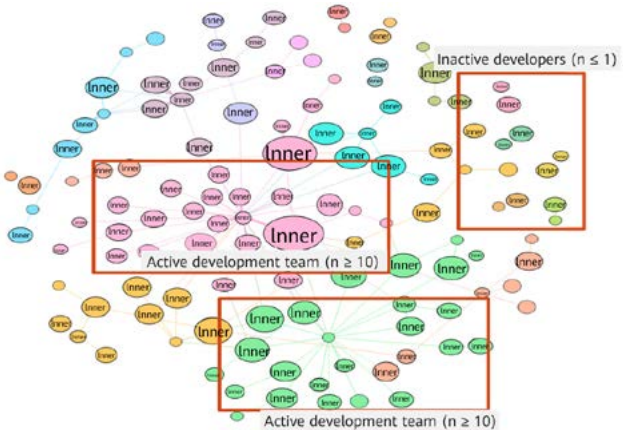


Figure 16 Collaboration relationship graph for the MindSpore community

successfully retained some developers who were planning to leave. In addition, by building a developer interest graph, we set up community security, front-end, data, and scientific computing SIGs in the MindSpore community, improving the overall activeness of the community.

3.4.2 Developer Experience Improvement Through the Developer NPS Measurement Model

This section provides a real-world application of the developer NPS measurement model in the openEuler community.

Table 1 defines the entire journey that a developer may experience after being recruited into the community through activities like meetup. Throughout the journey, each stop is provided with different developer portraits specific to the developer's role (newcomer/casual/regular/core). Then through NPS questionnaires, different questions are sent for different portraits to obtain clear feedback from developers. In this way, the community can detect its own problems and take measures to improve developers' loyalty to the community.

Table 1 Developer journey

Developer's Journey	Recruit	Use	Sign a CLA	Join SIGs	Create issues	Respond to issues	Close issues	Develop locally	Create PRs	CI	Review code	Incorporate PRs	Build versions	Release versions
Portraits	Newcomer portrait	User portrait	CLA signer portrait	SIG member portrait	Issue creator portrait	Issue responder portrait	Issue closer portrait	Developer portrait			Code reviewer portrait	Committer portrait	Publisher portrait	
Portrait Roles	Newcomer	Core Regular Casual	Core Regular Casual		Core Regular Casual			Core Regular Casual		Core Regular Casual	Core Regular Casual	Core Regular Casual	Core Regular Casual	
Developer Relationship Graph	Newcomer - Evangelist Mentor	User - Mentor	CLA signer - Mentor	SIG member - Maintainer	Issue creator - Maintainer	Issue responder - Maintainer/ Contributors	Issue closer - Maintainer	Developer - Maintainer Issue creator		Developer - Infra owner	Code reviewer - Developer Committer Maintainer	Committer - Committer Maintainer Issue creator	Publisher - Maintainer TC committee	
NPS Questionnaire	Newcomer NPS	User NPS	SIG NPS		Issue NPS			Local developer NPS		Builder NPS	Committer NPS		Publisher NPS	
	1. Community mind share 2. Recruitment channels	1. Satisfaction with use	1. Satisfaction with CLA signing	1. SIG satisfaction	1. Satisfaction with issue creation documents	1. Time to issue response 2. Satisfaction with issue support	1. Time to issue closure	1. Satisfaction with local development documents 2. Sample code 3. Development environment setup 4. Local commissioning 5. Local build, test, and deployment	1. Satisfaction with PR creation documents 2. PR-issue association rate	1. Efficiency 2. Satisfaction	1. Time to review response 2. Satisfaction with review	1. Time to PR closure	1. Satisfaction with published information 2. Satisfaction with versions	
NPS Roles	Promoter Passive Detractor	Promoter Passive Detractor	Promoter Passive Detractor		Promoter Passive Detractor			Promoter Passive Detractor		Promoter Passive Detractor	Promoter Passive Detractor		Promoter Passive Detractor	
Competing Community NPS	Carry out third-party NPS surveys. Through comparison with competing communities, identify relative NPSs and opportunities for improvement.													
Metrics	Set metrics based on the developer journey and NPS questionnaire.													
Feedback Mechanism	Based on the NPS questionnaire and real-time monitoring of the measurement platform, manage community operations and improve developers' loyalty to the community.													

4 Conclusion

Currently, an increasing number of enterprises are embracing open source and incorporating it into their development strategies. Two natural questions to ask are: Why must enterprises open source? And how should they open source? Based on the six elements of the open source strategy, this paper comprehensively reviewed the open source activities of enterprises from different perspectives, such as business, project, and ecosystem. It aims to help enterprises measure the health of open source communities, and build high-growth and high-loyalty open source communities in support of their open source strategies.

References

[1] Raymond and Eric S (1999), "The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary," O'Reilly Media. ISBN 1-56592-724-9.

[2] Richard Stallman, "FLOSS and FOSS," Accessed 9 October 2022.

[3] Bryan Che, "OpenSource policy"

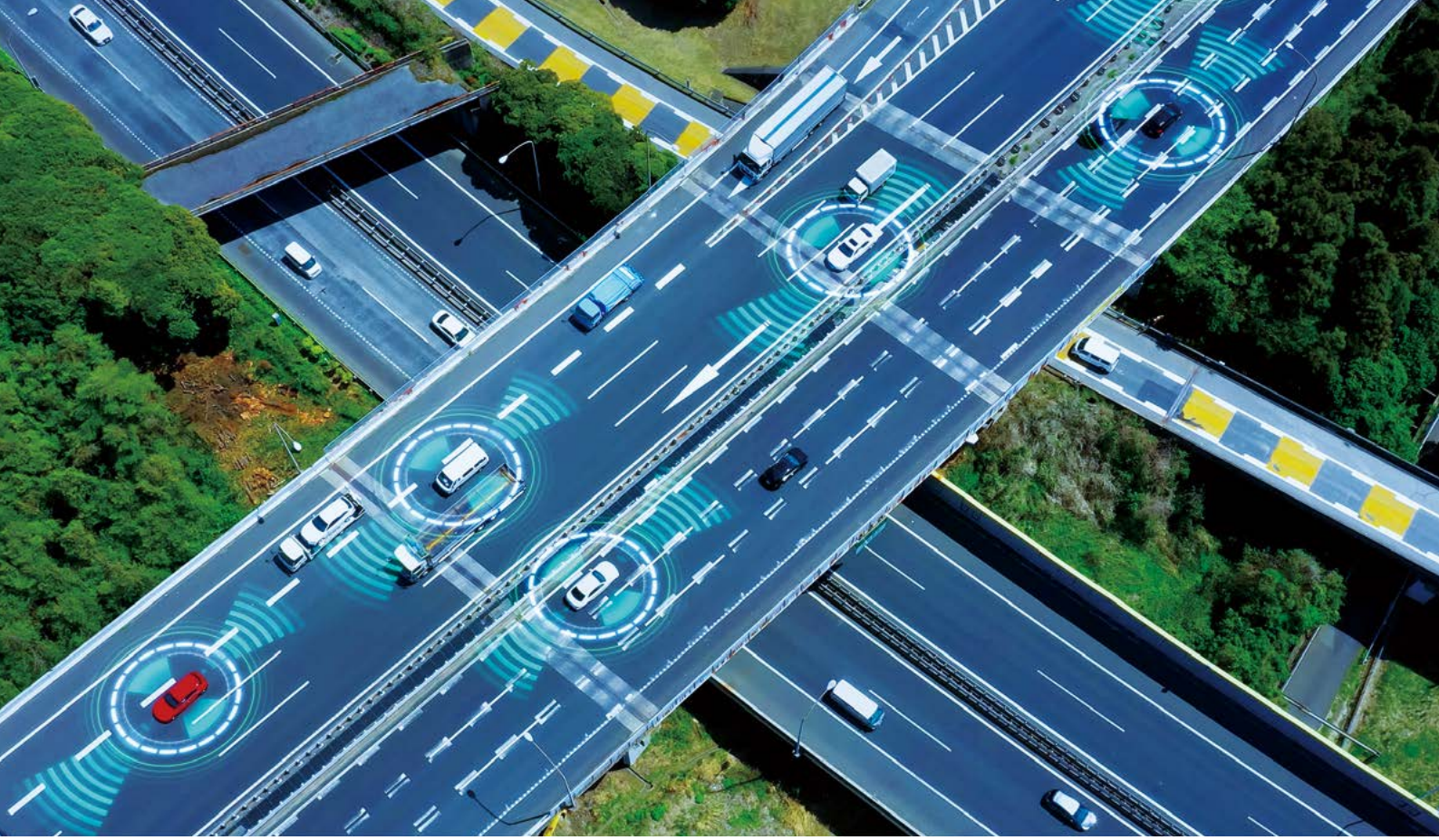
[4] Zhang Jian, Meng Xiangxin, Sun Hailong, Wang Xu, and Liu Xudong, "Data driven intelligent collaboration of software developers," Big Data Research [J], 202101: 76-93, 2021.

[5] Yang Cheng, Fan Qiang, Wang Tao, Yin Gang, and Wang Huaimin, "A multi-feature-based personalized recommendation approach for open-source repositories," Journal of Software, 2017, 28(6):1357-1372, 2017

[6] Jono Bacon (2019), "People powered," HarperCollins Focus. ISBN 9781400214884.

[7] Fred Reichheld and Rob Markel, "The ultimate question 2.0 (revised and expanded edition): How net promoter companies thrive in a customer-driven world," 2011

[8] Christine Abernathy, Chris Aniszczyk, Joe Beda, Sarah Novotny, and Gil Yehuda, "Measuring your open source program's success," Accessed 9 October 2022.



LiDAR Aided GNSS-RTK Positioning for Autonomous Vehicles

Han Gao [1], Weisong Wen [2], Li-Ta Hsu [2], Yongliang Wang [1]
[1] Riemann Lab
[2] Department of Aeronautical and Aviation Engineering, Hong Kong Polytechnic University

Abstract

Global navigation satellite system real-time kinematic (GNSS-RTK) positioning is an indispensable source for providing absolute positioning for autonomous driving vehicles (ADVs), due to its high accuracy when a fixed solution is obtained. However, its performance degrades dramatically in urban areas, due to signal blockage, reflection and diffraction from buildings. To fill this gap, this paper proposes a novel method to exclude the potential GNSS non-line-of-sight (NLOS) receptions using the local environment mapping generated from 3D LiDAR and inertial sensor, to further improve the GNSS-RTK in urban canyons. The local environment description, the 3D point cloud map (PCM), is built via LiDAR/inertial integration using the factor graph optimization. Then the potential GNSS NLOS receptions are detected and removed by using the 3D PCMs before GNSS-RTK positioning. Finally, the improved GNSS-RTK positioning is adopted to correct the drift of the 3D PCM derived from LiDAR/inertial integration. The effectiveness of the proposed method is verified by a test dataset in challenging urban environments collected by the automotive-grade low-cost GNSS receiver.

Keywords

GNSS, RTK, NLOS, LiDAR, urban canyon

1 Introduction

The global navigation satellite system real-time kinematic (GNSS-RTK) is widely used for high precision aerial mapping [1] and positioning for level-4 fully autonomous driving vehicles (ADVs) [2]. Typically, the GNSS-RTK positioning involves two steps: (1) The float solution is estimated based on the received GNSS measurements. (2) The integer ambiguity is resolved using least-squares algorithms (e.g LAMBDA [3]) based on the derived float solution as an initial guess. Centimeter-level positioning accuracy can be achieved based on the double-differential carrier and code measurements in the open area when the fixed solution is obtained. Unfortunately, the accuracy of GNSS-RTK is significantly degraded in urban canyons due to the NLOS and multipath receptions caused by GNSS signal reflection and blockage from surrounding buildings. In practice, the significantly degraded GNSS-RTK positioning accuracy in urban canyons is mainly due to the occurrence of GNSS NLOS receptions. A part of the received GNSS signals is significantly polluted involving large noise. According to our previous research in [4], the majority of the received GNSS signals can be multipath or NLOS receptions in highly urbanized areas. This degrades the accuracy of the float solution estimation that is based on the differential carrier and code measurements, making it more difficult to resolve for the integer ambiguity.

Numerous researches [5–7] have been conducted to improve the GNSS-RTK positioning performance in urban canyons in the past decades. The work in [6] proposed to employ multiple antennas to improve the robustness of GNSS-RTK against outlier measurements. However, the method relied on the percentage of the polluted GNSS signals (multipath or NLOS receptions). The recent work in [5] proposed to improve the GNSS-RTK in urban canyons by excluding polluted GNSS signals with the help of 3D building models. The increased fixed rate was obtained after selecting the line-of-sight (LOS) measurements. However, the satellite exclusion relies on the availability of accurate 3D building models and the initial guess of the GNSS receiver's position. Another research stream was to employ the additional sensors via sensor fusion. The integration of the GNSS-RTK and inertial measurement unit (IMU) was widely studied due to their complementariness. The work in [8] showed that with the help of better signal availability, high-grade dual-frequency multi-GNSS-RTK can achieve correct ambiguity integer fixing rates of 76.7% on a 1-hour drive along in a typical urban scenario. However, the overall performance relies heavily on the cost of the IMU sensor during the GNSS outage [9]. The work in [10] [11] proposed to tightly integrate the GNSS-RTK with the vision measurements in GNSS-challenged environments. Unfortunately, the vision measurements are sensitive to the illumination conditions and density of dynamic objects [12]. Moreover, the recovery of the scale of the visual measurement relies heavily on the quality of the GNSS measurements. Instead of using visual measurements, some previous researches in [4, 13–15] have proposed to continuously improve the GNSS single point positioning (SPP) in urban canyons by using 3D LiDAR sensors that are robust irrespective of illumination conditions. The 3D point clouds from the 3D LiDAR sensor are employed to describe the surrounding environment to further exclude [4] or correct GNSS NLOS receptions [13]. The latest work in [16] combines both the GNSS NLOS correction and remodeling to improve the urban GNSS SPP based on the incrementally built environmental description (3D PCMs). However, only the code measurements are applied and the potential of the carrier-phase measurements is not explored.

Inspired by related previous works on 3D LiDAR aided GNSS SPP [4, 13], in this paper, we propose to improve the GNSS-RTK positioning in urban canyons using a 3D LiDAR sensor by essentially solving the problem of GNSS-RTK caused by signal reflections and blockage. First, LiDAR/inertial odometry (LIO), which loosely integrates LiDAR and the IMU measurements using factor graph optimization (FGO) based on the recent work in [17], is performed to estimate the relative motion between two epochs and generate the 3D PCM, which is the so-called the local environmental description. Second, the potential GNSS NLOS satellites are detected and excluded with the help of the generated PCM based on the previous research work in [16]. Therefore, the problem of poor GNSS measurement quality is alleviated by excluding the potential GNSS NLOS receptions. Secondly, the float solution can be estimated based on the surviving GNSS satellite measurements. Then the LAMBDA algorithm is applied to perform the ambiguity resolution. Finally, the estimated fixed GNSS-RTK positioning solution is used with the LIO to further correct the drift of the 3D point clouds. In short, the proposed method effectively combines the complementariness of LIO (locally accurate in a short period and provides environmental description for GNSS NLOS detection) and GNSS-RTK (free of drift with globally referenced positioning but affected by the GNSS NLOS). The contributions of this paper are listed as follows:

(1) This paper proposes to use the LiDAR/inertial integration to detect and exclude the GNSS NLOS to further improve GNSS-RTK positioning. To the best of the author's knowledge, this is the first work that enables GNSS NLOS exclusion for GNSS-RTK positioning using LiDAR/inertial integration.

(2) This paper proposes to adopt the improved GNSS-RTK positioning to correct the drift of 3D point clouds, and therefore, improve the overall positioning accuracy.

(3) This paper evaluates the performance of the proposed method using the dataset collected using a low-cost GNSS receiver.

The remainder of this paper is organized as follows. An overview of the proposed method is given in Section 2. The generation of the local environmental description is elaborated in Section 3. In Section 4, the proposed NLOS detection and GNSS-RTK positioning are presented. An experiment is performed to evaluate the effectiveness of the proposed method using a dataset collected in urban canyons of Hong Kong, in Section 5. Finally, conclusions are drawn, and further work is presented in Section 6.

2 Overview of the Proposed Method

An overview of the method proposed in this paper is shown in Figure 1. The system consists of two parts: (1) The real-time environment description generation based on clouds from 3D LiDAR and an IMU, together with the correction from the GNSS-RTK solution. (2) The GNSS NLOS detection and exclusion based on the real-time environment description, and the GNSS-RTK positioning based on surviving satellites. In this paper, matrices are denoted in uppercase with bold letters. Vectors are denoted in lowercase with bold letters. Variable scalars are denoted in lowercase italic letters. Constant scalars are denoted in lowercase letters. Meanwhile, the state of the GNSS receiver and the position of satellites are all expressed in the east, north, up (ENU) coordinates.

To make the proposed pipeline clear, the following major notations are defined and followed by the rest of the paper.

- The pseudorange measurement received from a satellite s at a given epoch k is expressed as $\rho_{r,k}^{s}$. The subscript r and k denote the GNSS receiver and the time index, respectively. The superscript s denotes the index of the satellite.
- The carrier-phase measurement received from a satellite s at a given epoch k is expressed as $\psi_{r,k}^{s}$.
- The variable expressed in the earth-centered, earth-fixed (ECEF) frame or ENU frames is denoted by superscripts G and L. For example, the transformation of poses and positions from the ENU and the ECEF frame is defined as $\mathbf{T}_L^G = [\mathbf{R}_L^G, \mathbf{t}_L^G]$, where the $\mathbf{R}_L^G$ and the $\mathbf{t}_L^G$ denote the rotation and translation, respectively.
- The body frames of AHRS, LiDAR, and GNSS receiver are denoted by superscripts BI, BL, and BR. For example, $\mathbf{P}_k^{BL}$ denotes a frame of 3D point clouds from 3D LiDAR at epoch k.
- The extrinsic parameters between the GNSS receiver and the 3D LiDAR are denoted as $\mathbf{T}_{BL}^{BR} = [\mathbf{R}_{BL}^{BR}, \mathbf{t}_{BL}^{BR}]$. The extrinsic parameters between the IMU and the 3D LiDAR are denoted as $\mathbf{T}_{BL}^{BI} = [\mathbf{R}_{BL}^{BI}, \mathbf{t}_{BL}^{BI}]$.
- The position of the satellite s of ECEF frame at a given epoch k is expressed as $\boldsymbol{\rho}_k^s = [\rho_{k,x}^s, \rho_{k,y}^s, \rho_{k,z}^s]^{\mathrm{T}}$.
- The position of the GNSS receiver at a given epoch k in the ECEF and ENU frames are expressed as $\mathbf{p}_{r,k}^G = [p_{r,k,x}^G, p_{r,k,y}^G, p_{r,k,z}^G]^{\mathrm{T}}$ and $[p_{r,k,x}^L, p_{r,k,y}^L, p_{r,k,z}^L]^{\mathrm{T}}$. The rotation in the ENU frame is denoted as $\mathbf{R}_{r,k}^L = [\alpha_{r,k,x}^L, \beta_{r,k,y}^L, \gamma_{r,k,z}^L]^{\mathrm{T}}$.
- The clock bias of the GNSS receiver at a given epoch k is expressed as $\delta_{r,k}$, with the unit in meters. $\delta_{r,k}^s$ denotes the satellite clock bias in meters.

3 Local Environmental Description Generation

This section presents the methodology for the local environmental description generation based on LiDAR/inertial sensor, together with the correction from the GNSS-RTK. The LiDAR/inertial integration is first adopted to generate the 3D PCM using a local FGO (see Figure 1). Then the 3D PCM is employed to detect the GNSS NLOS receptions. However, the LiDAR/inertial integration is subjected to drift over time. Therefore, the improved GNSS-RTK positioning result is employed to correct the 3D PCM by integrating the GNSS-RTK with the LIO using a global FGO (see Figure 1). It should be noted that the PCM is denoted in the ENU frame. The extrinsic parameters between the ENU frame and the original of the LiDAR frame are calibrated using the first several fixed solutions from GNSS-RTK [18].

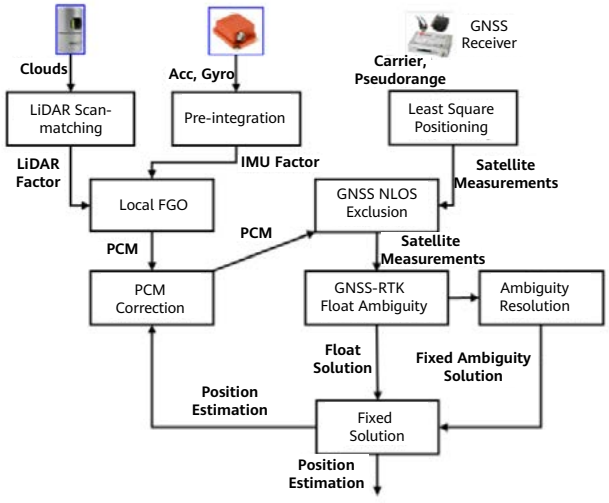


Figure 1 Overview of the proposed method. The inputs are the raw measurements from IMU, 3D LiDAR, and GNSS receiver. The output is the state estimation of the GNSS receiver.

3.1 LiDAR/Inertial Integration Using Local FGO

The LiDAR/inertial integration employed in this paper is based on the recent work in [17], where a sliding window-based loosely coupled integration using FGO is developed. Although this is not the major contribution of this paper, we still present it here concisely for completeness. The k-th state of the IMU under the world frame, that is the state of the IMU at the time when the k-th frame of the LiDAR point cloud is captured, can be written as:

$$\mathbf{x}_k^{BL^{(0)}} = \left[\mathbf{p}_{BI_k}^{BL^{(0)}}, \mathbf{v}_{BI_k}^{BL^{(0)}}, \mathbf{q}_{BI_k}^{BL^{(0)}}, \mathbf{ba}_k, \mathbf{bg}_k\right] \tag{1}$$

where $\mathbf{x}_k^{BL^{(0)}}$ consists of the position, velocity, rotation in quaternion form, accelerometer bias $\mathbf{ba}_k$ and gyroscope bias $\mathbf{bg}_k$. Therefore, the state set ($\mathbf{X}^{BL^{(0)}}$) inside a local window of the FGO can be denoted as:

$$\mathbf{X}^{BL^{(0)}} = \left[\mathbf{x}_0^{BL^{(0)}}, \mathbf{x}_1^{BL^{(0)}}, \cdots, \mathbf{x}_K^{BL^{(0)}}\right] \tag{2}$$

where the K denotes the size of the sliding window of the FGO.

3.1.1 IMU Preintegration Model

The IMU unit measures angular rate and the specific force of the system given in the IMU body frame. It's corrupted with the additive noise and a slowly varying bias of acceleration and gyroscope [19]:

$$\tilde{\mathbf{a}}^{BI} = \mathbf{R}_{BL^{(0)}}^{BI}(\mathbf{a}^{BL^{(0)}} - \mathbf{g}^{BL^{(0)}}) + \mathbf{ba} + \mathbf{n}_a \tag{3}$$

$$\tilde{\boldsymbol{\omega}}^{BI} = \boldsymbol{\omega}^{BI} + \mathbf{bg} + \mathbf{n}_g \tag{4}$$

$\tilde{\mathbf{a}}^{BI}$ is the raw IMU acceleration measurement in the IMU body frame, and $\mathbf{a}^{BL^{(0)}}$ is the noise-free acceleration of the system in the ENU frame. $\mathbf{g}^{BL^{(0)}}$ is the gravity in the world frame. $\mathbf{R}_{BL^{(0)}}^{BI} \in SO_3$ is the rotation matrix from the ENU frame to the IMU body frame. $\mathbf{ba}$ represents slowly varying acceleration bias, whose derivative is in Gaussian distribution. $\mathbf{n}_a \sim N(0, \boldsymbol{\sigma}_a^2)$ is the additive noise of the acceleration. $\tilde{\boldsymbol{\omega}}^{BI}$ is the raw IMU gyroscope measurements in the IMU body frame, $\boldsymbol{\omega}^{BI}$ is the noise-free rotation rate. $\mathbf{bg}$ is the bias of $\boldsymbol{\omega}^{BI}$. We also assume its derivative subjects to be in Gaussian distribution. $\mathbf{n}_g \sim N(0, \boldsymbol{\sigma}_g^2)$ is the additive noise of $\boldsymbol{\omega}^{BI}$. Considering the high-frequency properties of IMU measurements, the IMU pre-integration technique is adopted to stack a series of IMU measurements into a single factor [19]. Therefore, IMU pre-integration under the local frame $\{\cdot\}^{B_k}$ is formulated as:

$$\boldsymbol{\alpha}_{BI_{k+1}}^{BI_k} = \iint_{t_k}^{t_{k+1}} \mathbf{q}_{BI_t}^{BI_k}(\tilde{\mathbf{a}}^{BI_t} - \mathbf{ba}_k)\delta t^2 \tag{5}$$

$$\boldsymbol{\beta}_{BI_{k+1}}^{BI_k} = \int_{t_k}^{t_{k+1}} \mathbf{q}_{BI_t}^{BI_k}(\tilde{\mathbf{a}}^{BI_t} - \mathbf{ba}_k)\delta t \tag{6}$$

$$\boldsymbol{q}_{BI_{k+1}}^{BI_k} = \int_{t_k}^{t_{k+1}} \mathbf{q}_{BI_t}^{BI_k} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}(\tilde{\boldsymbol{\omega}}^{BI_t} - \mathbf{bg}_k) \end{bmatrix} \delta t \tag{7}$$

where $\boldsymbol{\alpha}_{BI_{k+1}}^{BI_k}$, $\boldsymbol{\beta}_{BI_{k+1}}^{BI_k}$, and $\boldsymbol{q}_{BI_{k+1}}^{BI_k}$ are the pre-integration of position, velocity, and rotation separately. In practice, IMU measurements are discrete and are synchronized with the LiDAR frames by linear interpolation. We use median integral as follows instead of continuous numerical integration shown above. BI_t and BI_{t+1} are assumed to be two consecutive time instants between two epochs BI_k and BI_{k+1} , and δt is the time interval between BI_t and BI_{t+1}.

The median integralacts as the measurements of relative motion between two IMU states to constrain them. The residual $\mathbf{r}_{IMU}(\mathbf{z}_{BI_{k+1}}^{BI_k}, \mathbf{X}^{BL^{(0)}})$ can be defined as [19]:

$$\mathbf{r}_{IMU}(\mathbf{z}_{BI_{k+1}}^{BI_k}, \mathbf{X}^{BL^{(0)}}) = \begin{bmatrix} (\mathbf{q}_{BI_k}^{BL^{(0)}})^{-1}(\mathbf{p}_{BI_{k+1}}^{BL^{(0)}} - \mathbf{p}_{BI_k}^{BL^{(0)}} - \mathbf{v}_{BI_k}^{BL^{(0)}}\delta t - \frac{1}{2}\mathbf{g}^{BL^{(0)}}\delta t^2) - \boldsymbol{\alpha}_{BI_{k+1}}^{BI_k} \\ 2\left[(\mathbf{q}_{BI_{k+1}}^{BI_k})^{-1} \otimes ((\mathbf{q}_{BI_k}^{BL^{(0)}})^{-1} \otimes \mathbf{q}_{BI_{k+1}}^{BL^{(0)}})\right]_{xyz} \\ (\mathbf{q}_{BI_k}^{BL^{(0)}})^{-1}(\mathbf{v}_{BI_{k+1}}^{BL^{(0)}} - \mathbf{v}_{BI_k}^{BL^{(0)}} - \mathbf{g}^{BL^{(0)}}\delta t) - \boldsymbol{\beta}_{BI_{k+1}}^{BI_k} \\ \mathbf{ba}_{k+1} - \mathbf{ba}_k \\ \mathbf{bg}_{k+1} - \mathbf{bg}_k \end{bmatrix} \tag{8}$$

where the operator $[\cdot]_{xyz}$ extracts the imaginary part of a quaternion. The $\mathbf{z}_{BI_{k+1}}^{BI_k}$ denotes the pre-integration

measurements which are a combination of $\boldsymbol{\alpha}_{BI_{k+1}}^{BI_k}$, $\boldsymbol{\beta}_{BI_{k+1}}^{BI_k}$, and $\boldsymbol{q}_{BI_{k+1}}^{BI_k}$.

3.1.2 LiDAR Scan-matching Measurement Model

The scan-matching method proposed in [20] is employed to derive the relative motion between consecutive frames of 3D point clouds. By accumulating the relative motion between consecutive frames of point clouds, pose estimation of LiDAR odometry at a given epoch k can be denoted as $\mathbf{T}_{BL_k}^{BL^{(0)}}$ satisfying the following [17]:

$$\mathbf{T}_{BL_k}^{BL^{(0)}} = \left[\tilde{\mathbf{p}}_{BL_k}^{BL^{(0)}}, \tilde{\mathbf{q}}_{BL_k}^{BL^{(0)}}\right] \tag{9}$$

Therefore, the residual $\mathbf{r}_L(\mathbf{T}_{L_k}^W, \mathbf{X})$ of the LiDAR scan-matching can be defined as the difference between $\mathbf{T}_{BL_k}^{BL^{(0)}}$ and $\mathbf{x}_k^{BL^{(0)}}$:

$$\mathbf{r}_L(\mathbf{T}_{BL_k}^{BL^{(0)}}, \mathbf{X}^{BL^{(0)}}) = \begin{bmatrix} \mathbf{p}_{BL_k}^{BL^{(0)}} - \tilde{\mathbf{p}}_{BL_k}^{BL^{(0)}} \\ 2\left[(\tilde{\mathbf{q}}_{BL_k}^{BL^{(0)}})^{-1} \otimes \mathbf{q}_{BL_k}^{BL^{(0)}}\right]_{xyz} \end{bmatrix} \tag{10}$$

The detailed derivation of the scan-matching can be found in our previous work in [17].

3.1.3 Solving Local FGO

Based on the residuals derived above, the combined objective function can be formulated as follows to optimize the state set $\mathbf{X}^{BL^{(0)}}$ inside the sliding window [21]:

$$\begin{aligned} \mathbf{X}^{BL^{(0)*}} = \arg\min \frac{1}{2}\{ & \sum_{k=0}^{K} \rho\left(\left\|\mathbf{r}_L(\mathbf{T}_{BL_k}^{BL^{(0)}}, \mathbf{X}^{BL^{(0)}})\right\|_{\mathbf{C}_{L_k}}^2\right) \\ & + \sum_{k=0}^{K-1} \rho\left(\left\|\mathbf{r}_{IMU}(\mathbf{z}_{BI_{k+1}}^{BI_k}, \mathbf{X}^{BL^{(0)}})\right\|_{\mathbf{C}_{BI_{k+1}}^{BI_k}}^2\right)\} \end{aligned} \tag{11}$$

where the $\mathbf{r}_L(\mathbf{T}_{BL_k}^{BL^{(0)}}, \mathbf{X}^{BL^{(0)}})$ represents the residuals of LiDAR scan-matching factor, $\mathbf{z}_{BI_{k+1}}^{BI_k}$ and $\mathbf{r}_{IMU}(\mathbf{z}_{BI_{k+1}}^{BI_k}, \mathbf{X}^{BL^{(0)}})$ represent the measurements and the residuals produced by the IMU pre-integration factor, respectively, $\mathbf{X}^{BL^{(0)*}}$ are the optimal states to be estimated. The $\mathbf{C}_{L_k}$ denotes the covariance matrix of the LiDAR scan-matching factor which is derived based on [17]. The $\mathbf{C}_{BI_{k+1}}^{BI_k}$ denotes the covariance matrix of the IMU pre-integration factor. The $\rho(*)$ denotes the robust loss function and the Cauchy kernel [22] is selected in this paper. Finally, the Ceres solver [23] is used to solve this nonlinear problem and we exploit the Levenberg-Marquardt (L-M) algorithm [24] to iteratively minimize the cost function (11). Therefore, the PCM can be acquired by accumulating the raw 3D point clouds inside the sliding window of local FGO based on the state set $\mathbf{X}^{BL^{(0)}}$, which is denoted as $\mathbf{M}_k^{BL^{(0)}}$.

3.2 LIO/GNSS-RTK Integration Using Global FGO

The global FGO integrates the pose estimation from the LIO presented in Section 3.1, and the GNSS-RTK positioning. Similar to the LIO, the state set inside the sliding window is optimized. The k-th state of the IMU frame in the ENU frame can be written as:

$$\mathbf{x}_k^L = \left[\mathbf{p}_{BI_k}^L, \mathbf{q}_{BI_k}^L\right], \tag{12}$$

where $\mathbf{x}_k^L$ consists of the position, and rotation in quaternion form in the ENU frame. Therefore, the state set ($\mathbf{X}_k^L$) inside a local sliding window of the FGO can be denoted as:

$$\mathbf{X}^L = \left[\mathbf{x}_0^L, \mathbf{x}_1^L, \cdots, \mathbf{x}_K^L\right], \tag{13}$$

where the K denotes the size of the sliding window of the FGO. Assuming the position estimation from the improved GNSS-RTK is denoted as follows:

$$\mathbf{z}_{r,k}^G = \left(p_{r,k,x}^G, p_{r,k,y}^G, p_{r,k,z}^G\right)^{\mathrm{T}} \tag{14}$$

where the $\mathbf{z}_{r,k}^G$ is denoted in the ECEF frame. Therefore, the residual of the $\mathbf{z}_{r,k}^G$ can be derived as follows:

$$\mathbf{r}_{\mathrm{GNSS}}(\mathbf{z}_{r,k}^G, \mathbf{X}^L) = \left[(\mathbf{T}_L^G)^{-1}(\mathbf{z}_{r,k}^G - \mathbf{t}_L^G) - \mathbf{p}_{BI_k}^L\right] \tag{15}$$

where the $\mathbf{r}_{\mathrm{GNSS}}(\mathbf{z}_{r,k}^G, \mathbf{X}^L)$ denotes the residual associated with $\mathbf{z}_{r,k}^G$. Based on the LIO, the observation at a given epoch k is $\mathbf{x}_k^{BL^{(0)}}$ and $\mathbf{x}_{k-1}^{BL^{(0)}}$ at epoch $k-1$. The residual can be formulated as follows:

$$\begin{aligned} & \mathbf{r}_{\mathrm{LIO}}(\mathbf{x}_{k-1}^{BL^{(0)}}, \mathbf{x}_k^{BL^{(0)}}, \mathbf{x}_{k-1}^L, \mathbf{x}_k^L) = \\ & \begin{bmatrix} (\mathbf{P}_1 - \mathbf{P}_2) - (\mathbf{p}_{BI_k}^L - \mathbf{p}_{BI_{k-1}}^L) \\ 2\left[((\mathbf{R}_2)^{-1}\mathbf{R}_1) \otimes (\mathbf{p}_{BI_{k-1}}^L \otimes \mathbf{p}_{BI_k}^L)\right]_{xyz} \end{bmatrix} \end{aligned} \tag{16}$$

with

$$\begin{cases} \mathbf{P}_1 = (\mathbf{p}^L_{BL^{(0)}})^{-1}((\mathbf{R}^{BI}_{BL})^{-1}(\mathbf{p}^{BL^{(0)}}_{BL_k} - \mathbf{t}^{BI}_{BL}) - \mathbf{t}^L_{BL^{(0)}}) \\ \mathbf{P}_2 = (\mathbf{p}^L_{BL^{(0)}})^{-1}((\mathbf{R}^{BI}_{BL})^{-1}(\mathbf{p}^{BL^{(0)}}_{BL_{k-1}} - \mathbf{t}^{BI}_{BL}) - \mathbf{t}^L_{BL^{(0)}}) \\ \mathbf{R}_1 = (\mathbf{q}^L_{BL^{(0)}})^{-1}\mathbf{q}^{BL^{(0)}}_{BL_k}(\mathbf{R}^{BI}_{BL})^{-1} \\ \mathbf{R}_2 = (\mathbf{q}^L_{BL^{(0)}})^{-1}\mathbf{q}^{BL^{(0)}}_{BL_{k-1}}(\mathbf{R}^{BI}_{BL})^{-1} \end{cases} \tag{17}$$

where the $\mathbf{P}_1$ and $\mathbf{P}_2$ are defined for better representation. Based on the residuals derived above, the combined objective function can be formulated as follows to optimize the state set $\mathbf{X}^L$ inside the sliding window [21]:

$$\begin{aligned} \mathbf{X}^{L^*} = \arg\min \frac{1}{2}\{ \sum_{k=0}^{K} \rho(\left\|\mathbf{r}_{\mathrm{GNSS}}(\mathbf{z}^G_{r,k}, \mathbf{X}^L)\right\|^2_{\mathbf{C}_{\mathrm{GNSS}}}) \\ + \sum_{k=0}^{K-1} \rho(\left\|(\mathbf{x}^{BL^{(0)}}_{k-1}, \mathbf{x}^{BL^{(0)}}_{k}, \mathbf{x}^L_{k-1}, \mathbf{x}^L_k)\right\|^2_{\mathbf{C}_{\mathrm{LIO}}}) \} \end{aligned} \tag{18}$$

where the $\mathbf{r}_{\mathrm{GNSS}}(\mathbf{z}^G_{r,k}, \mathbf{X}^L)$ represents the residuals of the GNSS factor and $\mathbf{r}_{\mathrm{LIO}}(\mathbf{x}^{BL^{(0)}}_{k-1}, \mathbf{x}^{BL^{(0)}}_{k}, \mathbf{x}^L_{k-1}, \mathbf{x}^L_k)$ represents the residual of the LIO factor, respectively, $\mathbf{X}^{L^*}$ are the optimal states to be estimated. The $\mathbf{C}_{GNSS}$ denotes the covariance matrix of the GNSS factor. The $\mathbf{C}_{LIO}$ denotes the covariance matrix of the LIO factor. Similar to the local FGO, the Equation (18) is also solved using the Levenberg-Marquardt (L-M) algorithm [24] via Ceres-solver [23]. Therefore, the PCM can be corrected by accumulating the raw 3D point clouds inside the sliding window of global FGO based on the state set $\mathbf{X}^L$, which is denoted as $\mathbf{M}^L_k$. Different from our previous work in [16], the generated PCM is free of drift with the help of the integration of GNSS-RTK positioning, which is one of the contributions of this paper.

4 NLOS Exclusion and GNSS-RTK Positioning

4.1 GNSS NLOS Exclusion Based on PCM

The pseudorange measurement from the GNSS receiver, $\rho^s_{r,k}$, is denoted as follows [25].

$$\rho^s_{r,k} = r^s_{r,k} + c(\delta_{r,k} - \delta^s_{r,k}) + I^s_{r,k} + T^s_{r,k} + \varepsilon^s_{r,k} \tag{19}$$

where $r^s_{r,k}$ is the geometric range between the satellite and the GNSS receiver. $I^s_{r,k}$ represents the ionospheric delay distance; $T^s_{r,k}$ indicates the tropospheric delay distance. $\varepsilon^s_{r,k}$ represents the errors caused by the multipath effects, NLOS receptions, receiver noise, and antenna phase-related noise. The atmosphere effects ($T^s_{r,k}$ and $I^s_{r,k}$) are compensated using the conventional models (Saastamoinen and Klobuchar models, respectively) presented in RTKLIB [26]. Given the PCM in the ENU frame, satellite elevation and azimuth angles which can be calculated using least squares estimation based on pseudorange measurements, the GNSS NLOS can be detected using the fast searching method proposed in [16]. An illustration of the GNSS NLOS receptions detection is shown in Figure 2.

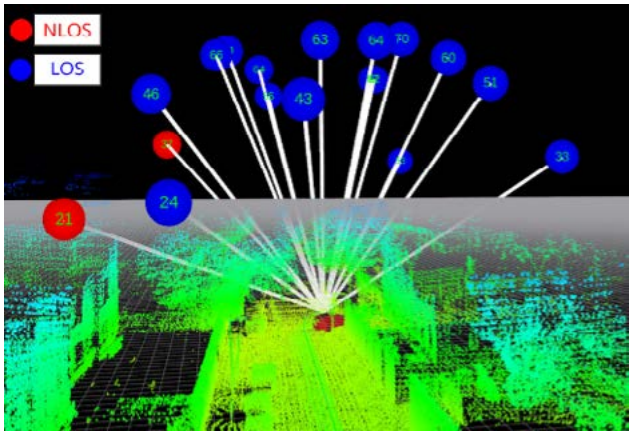


Figure 2 Demonstration of NLOS detection based on PCM. The red circles denote the GNSS NLOS satellites. The blue circles denote the LOS measurements. The number inside the circles denotes the elevation angle of the satellites.

Assume the received satellite set at epoch k is denoted as $\mathbf{SV}_{r,k} = \{SV^1_{r,k}, \cdots, SV^i_{r,k}, \cdots, SV^N_{r,k}\}$ which involves both the LOS and NLOS satellites. The variable N denotes the number of satellites received at epoch k. The $SV^i_{r,k}$ denotes the measurements of satellite i. After the GNSS NLOS detection and exclusion, the remaining satellite set is denoted as $\mathbf{SSV}_{r,k} = \{SV^1_{r,k}, \cdots, SV^i_{r,k}, \cdots, SV^M_{r,k}\}$ where the variable M denotes the number of remaining satellites.

4.2 GNSS-RTK Positioning Based on Surviving Satellites

In terms of the measurements from the GNSS receiver, each carrier-phase measurement, $\psi^s_{r,k}$, is written as follows [25].

$$\lambda\psi^s_{r,k} = r^s_{r,k} + c(\delta_{r,k} - \delta^s_{r,k}) + I^s_{r,k} + T^s_{r,k} + \lambda B^s_{r,k} + \delta\psi^s_{r,k} + \varepsilon^s_{r,k} \tag{20}$$

The variable λ denotes the carrier wavelength. The variable $\delta\psi^s_{r,k}$ denotes the carrier-phase correction term including antenna phase offsets and variations, station displacement by earth tides, phase windup effect, and relativity correction on the satellite clock. $\varepsilon^s_{r,k}$ represents the errors caused by the multipath effects, NLOS receptions, receiver noise, and antenna delay. $B^s_{r,k}$ is the carrier-phase bias, which

is calculated in the following equation. The detailed formulation of the carrier-phase correction can be found in [26].

$$B^s_{r,k} = \psi_{r,0,k} - \psi^s_{0,k} + N^s_{r,k} \tag{21}$$

The variable $\psi_{r,0,k}$ represents the initial phase of the receiver local oscillator. Similarly, the $\psi^s_{0,k}$ stands for the initial phase of the transmitted navigation signal from the satellite. The variable $N^s_{r,k}$ denotes the carrier-phase integer ambiguity.

The double difference (DD) pseudorange measurement ($\rho^s_{DD,k}$) of satellite s is formulated as follows:

$$\rho^s_{DD,k} = (\rho^s_{r,k} - \rho^s_{b,k}) - (\rho^w_{r,k} - \rho^w_{b,k}) \tag{22}$$

The variable $\rho^w_{b,k}$ and $\rho^s_{b,k}$ stands for the pseudorange measurements received by the reference station which is denoted by the subscript b. Generally, the satellite with the highest elevation angle tends to involve the lowest multipath and NLOS errors. Therefore, the satellite w, with the highest elevation angle, is selected as the master satellite. After applying the DD process to the pseudorange measurements, the derived $\rho^s_{DD,k}$ is free of the clock bias and atmosphere effects [26]. Similarly, the DD carrier-phase measurement ($\psi^s_{DD,k}$) of satellite s is formulated as follows:

$$\psi^s_{DD,k} = (\psi^s_{r,k} - \psi^s_{b,k}) - (\psi^w_{r,k} - \psi^w_{b,k}) \tag{23}$$

The variables $\psi^s_{b,k}$ and $\psi^w_{b,k}$ stand for the carrier-phase measurements received by the reference station. Similarly, the clock bias and atmosphere effects are waived from $\psi^s_{DD,k}$. The $\psi^s_{DD,k}$ involves the DD ambiguity [26], which is to be estimated.

The float solution of GNSS-RTK can be represented as follows:

$$\mathbf{x}^{float}_{r,k} = (\mathbf{p}^G_{r,k}, \Delta N^1_{rb,k}, \Delta N^2_{rb,k}, \cdots, \Delta N^{M-1}_{rb,k})^{\mathrm{T}} \tag{24}$$

where the variable $\mathbf{x}^{float}_{r,k}$ denotes the state of the GNSS receiver at epoch k which consists of position ($\mathbf{p}^G_{r,k}$) in the ECEF frame, and the DD ambiguities. The variable $\Delta N^{M-1}_{rb,k}$ denotes the DD carrier-phase ambiguity bias corresponding to satellite $M-1$. In other words, each DD carrier-phase measurement involves a specific ambiguity bias. Therefore, the observation model for the DD pseudorange measurement ($\rho^s_{DD,k}$) is expressed as follows:

$$\rho^s_{DD,k} = \mathbf{h}^s_{DD,\rho,k}(\mathbf{x}_{r,k}, \mathbf{p}^s_k, \mathbf{p}^w_k, \mathbf{p}_b) + \omega^s_{DD,\rho,k} \tag{25}$$

$$\begin{aligned}\mathbf{h}^s_{DD,\rho,k}(\mathbf{x}_{r,k}, \mathbf{p}^s_k, \mathbf{p}^w_k, \mathbf{p}_b) = &(\|\mathbf{x}_{r,k} - \mathbf{p}^s_k\| - \|\mathbf{p}_b - \mathbf{p}^s_k\|) \\ &- (\|\mathbf{x}_{r,k} - \mathbf{p}^w_k\| - \|\mathbf{p}_b - \mathbf{p}^w_k\|)\end{aligned} \tag{26}$$

the variable $\omega^s_{DD,\rho,k}$ denotes the noise associated with the $\rho^s_{DD,k}$. The function $\mathbf{h}^s_{DD,\rho,k}(*)$ denotes the observation function connecting the state of the GNSS receiver and the DD measurement $\rho^s_{DD,k}$. Therefore, the error factor for the DD pseudorange measurement is as follows:

$$\begin{aligned}&\left\|e^s_{DD,\rho,k}\right\|^2_{\Sigma^s_{DD,\rho,k}} \\ &= \left\|\rho^s_{DD,k} - \mathbf{h}^s_{DD,\rho,k}(\mathbf{x}_{r,k}, \mathbf{p}^s_k, \mathbf{p}^w_k, \mathbf{p}_b)\right\|^2_{\Sigma^s_{DD,\rho,k}}\end{aligned} \tag{27}$$

The variable $\Sigma^s_{DD,\rho,k}$ stands for the covariance associated with the $\rho^s_{DD,k}$ which is calculated based on the elevation angle and Signal-to-Noise Ratio (SNR) of the received satellite measurement [27]. Similarly, the observation model for the DD carrier-phase measurement is expressed as follows:

$$\psi^s_{DD,k} = \mathbf{h}^s_{DD,\psi,k}(\mathbf{x}_{r,k}, \mathbf{p}^s_k, \mathbf{p}^w_k, \mathbf{p}_b) + \omega^s_{DD,\psi,k} \tag{28}$$

$$\begin{aligned}\mathbf{h}^s_{DD,\psi,k}(\mathbf{x}_{r,k}, \mathbf{p}^s_k, \mathbf{p}^w_k, \mathbf{p}_b) = &(\|\mathbf{x}_{r,k} - \mathbf{p}^s_k\| - \|\mathbf{p}_b - \mathbf{p}^s_k\|) \\ &- (\|\mathbf{x}_{r,k} - \mathbf{p}^w_k\| - \|\mathbf{p}_b - \mathbf{p}^w_k\|) + \Delta N^s_{rb,k}\end{aligned} \tag{29}$$

The variable $\omega^s_{DD,\psi,k}$ denotes the noise associated with the $\psi^s_{DD,k}$. The variable $\Delta N^s_{rb,k}$ denotes the DD ambiguity of the carrier-phase measurement. Therefore, the error factor for the DD carrier-phase measurement is as follows:

$$\begin{aligned}&\left\|e^s_{DD,\psi,k}\right\|^2_{\Sigma^s_{DD,\psi,k}} \\ &= \left\|\psi^s_{DD,k} - \mathbf{h}^s_{DD,\psi,k}(\mathbf{x}_{r,k}, \mathbf{p}^s_k, \mathbf{p}^w_k, \mathbf{p}_b)\right\|^2_{\Sigma^s_{DD,\psi,k}}\end{aligned} \tag{30}$$

The variable $\Sigma^s_{DD,\psi,k}$ stands for the covariance associated with the $\psi^s_{DD,k}$. Therefore, the objective function for the float solution estimation of GNSS-RTK is formulated as follows:

$$\mathbf{x}^{float*}_{r,k} = \mathbf{arg\ min} \sum_{s,k} (\left\|e^s_{DD,\psi,k}\right\|^2_{\Sigma^s_{DD,\psi,k}} + \left\|e^s_{DD,\rho,k}\right\|^2_{\Sigma^s_{DD,\rho,k}}) \tag{31}$$

The variable $\mathbf{x}^{float*}_{r,k}$ denotes the optimal estimation of the float solution. Therefore, the float solution for GNSS-RTK at the current epoch can be obtained by solving the above objective function (31) via Ceres-solver [23]. After obtaining the float solution of the GNSS-RTK, the ambiguity resolution algorithm is used to estimate the fixed solution. The variable $\Delta N^s_{r,k}$ should be an integer value when the carrier-phase measurement is free from noise. This paper makes use of the popular LAMBDA algorithm [28] to solve

the integer ambiguity resolution problem and the resolved fixed solution is denoted as $\mathbf{x}_{r,k}^{fix}$. Then the fixed solution is fed to the global FGO to correct the drift in the PCM presented in Section 3.2.

5 Experimental Results and Discussion

5.1 Experimental Setup

To verify the effectiveness of the proposed method, we collect the dataset in a typical urban scenario in Hong Kong. Figure 3 shows the data collection vehicle installed with multiple sensors. The test scenario is shown on the bottom right of Figure 3. It is a challenging environment for the GNSS-RTK positioning, with tall buildings and trees on either sides of the street.

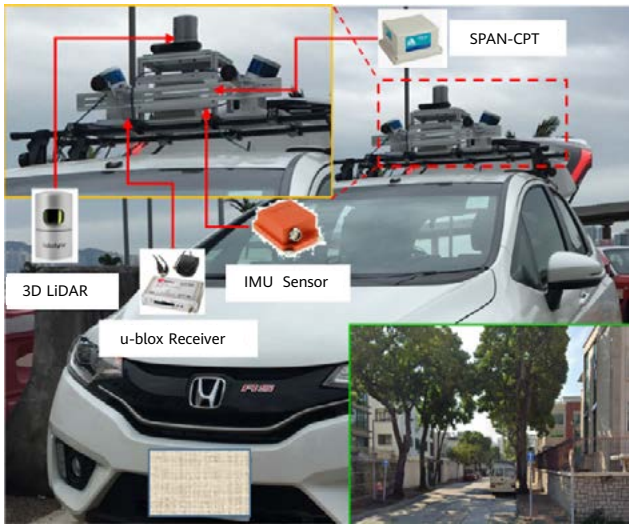


Figure 3 Illustration of the data collection vehicle and tested scenarios

During the experiment, a u-blox M8T GNSS receiver was used to collect raw GPS/BeiDou measurements at a frequency of 1 Hz. A 3D LiDAR sensor (Velodyne 32) was deployed to collect raw 3D point clouds at a frequency of 10 Hz. The Xsens Ti-10 IMU was used to collect data at a frequency of 200 Hz. In addition, the NovAtel SPAN-CPT, a GNSS (GPS, GLONASS, and Beidou) RTK/INS (fiber-optic gyroscopes, FOG) integrated navigation system was used to provide ground truth of positioning. The gyro bias in-run stability of the FOG is 1 degree per hour, and its random walk is 0.067 degrees per hour. The baseline between the rover and the GNSS base station is about 5 km. All the data was collected and synchronized using a robot operation system (ROS) [29]. The coordinate systems between all the sensors were calibrated before the experiments.

We analyzed the performance of GNSS-RTK positioning by comparing several methods, as shown below. The objective of this analysis was to validate the effectiveness of the proposed method in improving the GNSS-RTK positioning. The accuracy is evaluated in the ENU frame by selecting the first point as the reference position.

(a) GNSS-RTK: conventional GNSS-RTK positioning solution [26] via the raw measurements from the u-blox M8T receiver. The ambiguity is solved by an epoch-by-epoch basis.

(b) u-blox receiver: commercial GNSS positioning solution output from the u-blox receiver M8T receiver, without the correction from the reference stations.

(c) GNSS-RTK-C: GNSS-RTK positioning solution aided by GNSS NLOS exclusion using the proposed method, via the raw measurements from the u-blox M8T receiver. The ambiguity is solved by an epoch-by-epoch basis.

5.2 Performance Evaluation in an Urban Canyon

The results of the preceding three methods are compared in Table 1. The first row shows the number of satellites being used in the positioning estimation. A mean number of 17 satellites are received for GNSS positioning. After applying the proposed GNSS NLOS exclusion, the mean number of satellites decreases to 15.8. The second column shows the 2D positioning error of the u-blox receiver. The positioning result is based on standard NMEA messages from the u-blox receiver. A mean error of 6.25 meters was obtained, with a standard deviation of 7.31 meters. The maximum error reached 38.53 meters. The GNSS solution was available throughout the experiment. The third column shows the conventional GNSS-RTK positioning result. The mean error decreases to 2.43 meters after using both the raw measurements from the u-blox receiver and the reference station. The standard deviation (STD) and the maximum error decrease to 1.16 meters and 7.20 meters, respectively. We can see that the fixed rate is only 1.0% due to the poor measurement quality. With the help of the proposed method by excluding the GNSS NLOS measurements, the mean error decreases to 1.95 meters with a standard deviation of 1.003 meters. However, the maximum error increases from 7.20 (GNSS-RTK) to 12.40 meters (GNSS-RTK-C). Fortunately, the fixed rate increases slightly to 1.6%

after excluding the NLOS receptions. The improved GNSS positioning results demonstrate the effectiveness of the proposed method in mitigating the effects of NLOS signals.

Table 1 Positioning Performance Comparison

Items	u-blox	GNSS-RTK	GNSS-RTK-C
Mean satellite number	17	17	15.8
Mean error	6.25m	2.43m	1.95m
STD	7.31m	1.16m	1.003m
Maximum error	38.53m	7.20m	12.40m
Fixed rate	N/A	1.0%	1.6%

The trajectories of the evaluated three methods together with the ground truth are shown in Figure 4. The positioning errors throughout the experiment are shown in Figure 5. Interestingly, we can see that near epoch A annotated by a circle in Figure 5, the positioning error arising from the proposed method is significantly larger than the conventional GNSS-RTK which is due to the excessive exclusion of the GNSS NLOS. The scenario near epoch A is shown in the middle of Figure 4 which involves dense foliage. The bottom panel of Figure 5 shows the number of satellites being excluded. Four of the satellites are detected as NLOS using the generated PCM. As a result, the geometry of the satellite distribution is significantly distorted, leading to increased positioning error. Similar phenomenon were also found in previous literature [4, 30, 31].

In summary, the proposed method (GNSS-RTK-C) effectively detects and excludes the potential GNSS NLOS, leading to improved performance, compared with the conventional method (GNSS-RTK). However, the fixed rate is still limited at 1.0% for GNSS-RTK and 1.6% for GNSS-RTK-C. The major problem is caused by the poor geometry of the satellite distribution due to the proposed GNSS NLOS exclusion.

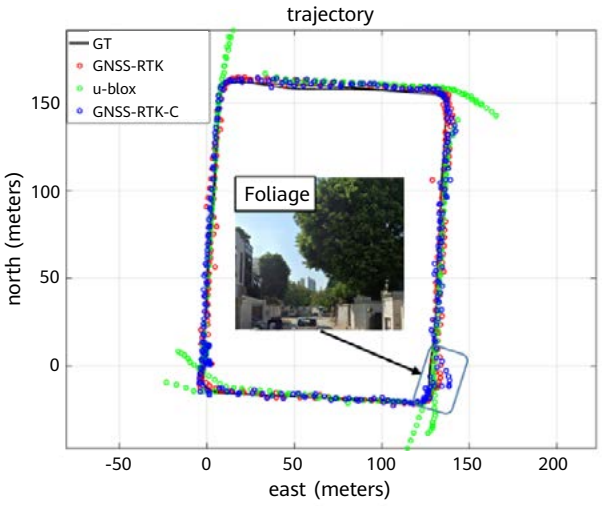


Figure 4 Trajectories of the evaluated methods. The black curve denotes the ground truth (GT). The red, green, and blue curves denote the solutions from GNSS-RTK, u-blox receiver, and GNSS-RTK-C, respectively.

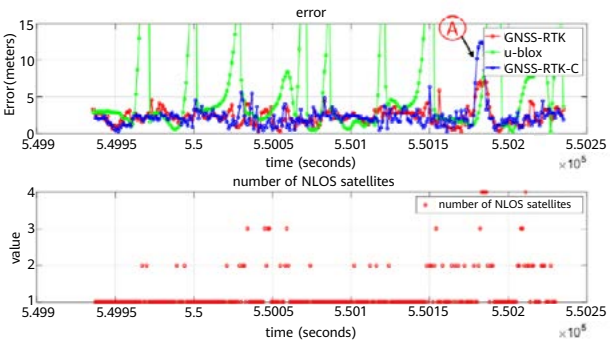


Figure 5 The top panel shows the errors of the evaluated methods. The red, green, and blue curves denote the solutions from GNSS-RTK, u-blox receiver, and GNSS-RTK-C, respectively. The bottom panel shows the number of excluded GNSS NLOS satellites during the experiment.

6 Conclusion

GNSS-RTK positioning is currently still the indispensable source for autonomous driving localization which requires accurate and absolute positioning. Unfortunately, the application of GNSS-RTK positioning in urban canyons is still limited due to the signal reflection and poor satellite geometry. This paper opens a new window for improving the GNSS-RTK by detecting and excluding the potential GNSS NLOS using onboard sensing (LiDAR/inertial integration). The accuracy of the GNSS-RTK is improved from 2.43 to 1.95 meters in the evaluated dataset with the help of the proposed NLOS exclusion.

According to the results, the fixed rate of GNSS-RTK is still limited in the evaluated dataset due to the poor satellite geometry after the GNSS NLOS exclusion. In the future, more environmental features will be exploited as pseudo-satellites to obtain better satellite geometry, in order to further increase the integer ambiguity fix rate of the GNSS-RTK in urban canyons. Also, multiple 3D LiDARs will be employed to generate more comprehensive and detailed PCM to improve the FOV of the environment reconstruction.

References

[1] P. J. Teunissen and A. Kleusberg, "GPS for geodesy," *Springer Science & Business Media*, 2012.

[2] G. Wan et al., "Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes," in 2018 *IEEE International Conference on Robotics and Automation (ICRA)*, 2018: IEEE, pp. 4670-4677.

[3] P. J. Teunissen, "Least-squares estimation of the integer GPS ambiguities," in I*nvited lecture, section IV theory and methodology*, IAG general meeting, Beijing, China, 1993.

[4] W. Wen, G. Zhang, and L. T. Hsu, "Correcting NLOS by 3D LiDAR and building height to improve GNSS single point positioning," *Navigation*, vol. 66, no. 4, pp. 705-718, 2019.

[5] R. Furukawa, N. Kubo, and A. El-Mowafy, "Prediction of RTK-GNSS performance in urban environments using a 3D model and continuous LOS method," in *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation*, 2020, pp. 763-771.

[6] P. Fan, W. Li, X. Cui, and M. Lu, "Precise and robust RTK-GNSS positioning in urban environments with dual-antenna configuration," *Sensors*, vol. 19, no. 16, p. 3586, 2019.

[7] T. Li, H. Zhang, Z. Gao, Q. Chen, and X. Niu, "High-accuracy positioning in urban environments using single-frequency multi-GNSS RTK/MEMS-IMU integration," *Remote Sensing*, vol. 10, no. 2, p. 205, 2018.

[8] G. C. Chung, S. T. Su, and M. Y. Alias, "Adaptive windowed statistical selection rake for long ultra-wideband multipath channels," (in English), *Wireless Pers Commun*, vol. 98, no. 1, pp. 453-466, Jan 2018, doi: 10.1007/s11277-017-4878-8.

[9] H. T. Zhang, "Performance comparison of kinematic GPS integrated with different tactical grade IMUs," *CALGARY, ALBERTA: THE UNIVERSITY OF CALGARY*, 2006.

[10] P. Henkel, A. Blum, and C. Günther, "Precise RTK positioning with GNSS, INS, Barometer and Vision," in *Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*, 2017, pp. 2290-2303.

[11] T. Li, H. Zhang, Z. Gao, X. Niu, and N. El-Sheimy, "Tight fusion of a monocular camera, MEMS-IMU, and single-frequency multi-GNSS RTK for precise navigation in GNSS-challenged environments," *Remote Sensing*, vol. 11, no. 6, p. 610, 2019.

[12] X. Bai, W. Wen, and L.-T. Hsu, "Performance analysis of visual/inertial integrated positioning in typical urban scenarios of Hong Kong," in *Proceedings of 2019 Asian-Pacific Conference on Aerospace Technology and Science*, 2019.

[13] W. Wen, G. Zhang, and L.-T. Hsu, "GNSS NLOS exclusion based on dynamic object detection using LiDAR point cloud," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[14] G. Zhang, Weisong Wen, and L.-T. Hsu, "Correcting GNSS NLOS by 3D LiDAR and building height," *ION GNSS+ 2018*, Miami, Florida, USA, 2018.

[15] W. Wen, G. Zhang, and L.-T. Hsu, "Exclusion of GNSS NLOS receptions caused by dynamic objects in heavy traffic urban scenarios using real-time 3D point cloud: An approach without 3D maps," in *Position, Location and Navigation Symposium (PLANS), 2018 IEEE/ION*, 2018: IEEE, pp. 158-165.

[16] W. Wen, "3D LiDAR aided GNSS and its tightly coupled integration with INS via factor graph optimization," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, 2020, pp. 1649-1672.

[17] J. Zhang, W. Wen, F. Huang, and L.-T. Hsu, "Loosely coupled Lidar-inertial odometry for urban positioning and mapping (to be submitted)," *Remote Sensing*, 2021.

[18] W. Wen, G. Zhang, and L.-T. Hsu, "Object-detection-aided GNSS and its integration with Lidar in highly urbanized areas," *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 3, pp. 53-69, 2020.

[19] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual--inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1-21, 2016.

[20] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014, vol. 2, no. 9.

[21] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations Trends in Robotics*, vol. 6, no. 1-2, pp. 1-139, 2017.

[22] Z. Zhang, "Parameter estimation techniques: a tutorial with application to conic fitting," *Image vision Computing*, vol. 15, no. 1, pp. 59-76, 1997.

[23] S. Agarwal and K. Mierle. "Ceres Solver." http://ceres-solver.org (accessed 6 January 2021).

[24] J. J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory," in *Numerical analysis: Springer*, 1978, pp. 105-116.

[25] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.

[26] T. Takasu and A. Yasuda, "Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB," in *International symposium on GPS/GNSS*, 2009: International Convention Center Jeju Korea, pp. 4-6.

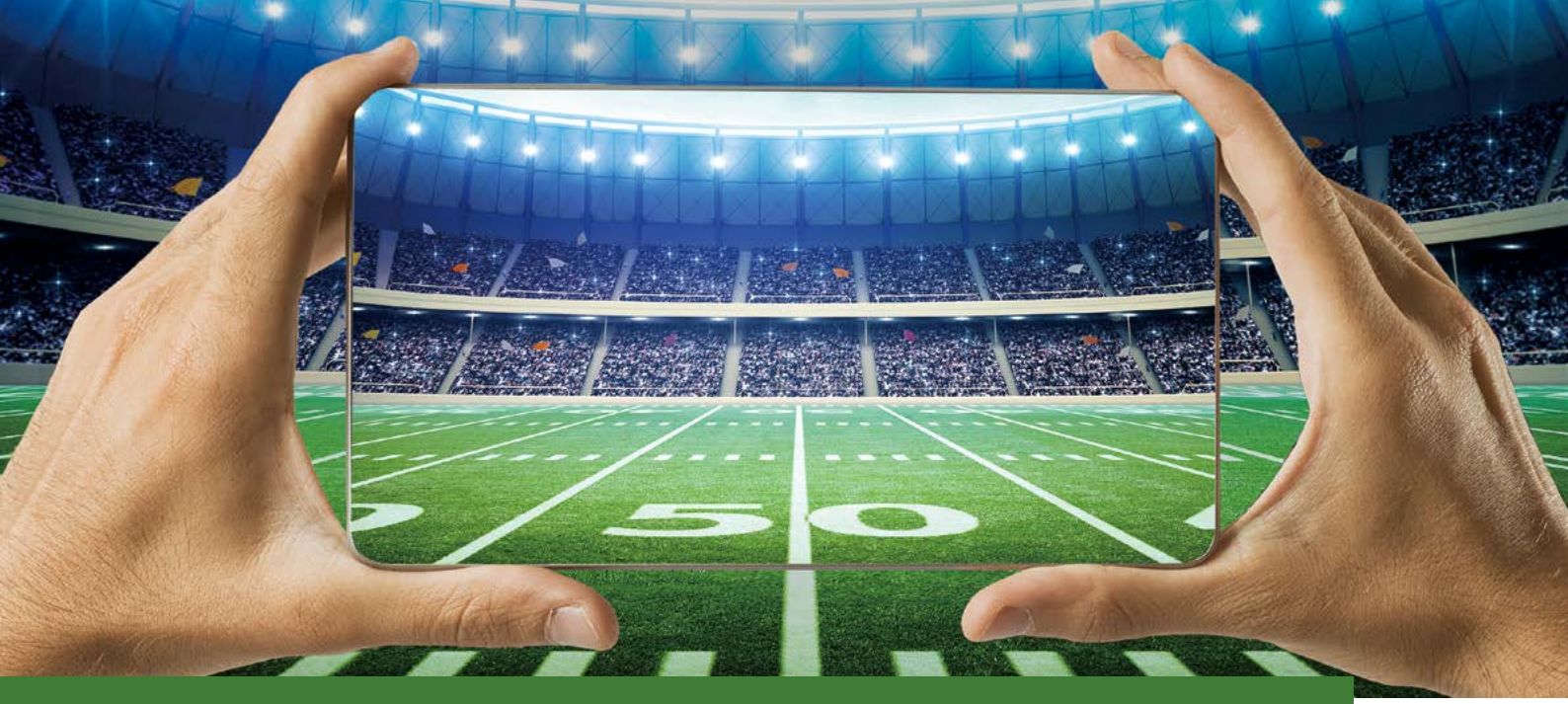
[27] A. M. Herrera, H. F. Suhandri, E. Realini, M. Reguzzoni, and M. C. de Lacy, "goGPS: open-source MATLAB software," *GPS Solution*, vol. 20, no. 3, pp. 595-603, 2016.

[28] P. Teunissen, "Theory of integer equivariant estimation with application to GNSS," *Journal of Geodesy*, vol. 77, no. 7-8, pp. 402-410, 2003.

[29] M. Quigley et al., "ROS: an open-source Robot Operating System," in I*CRA workshop on open source software*, 2009, vol. 3, no. 3.2: Kobe, Japan, p. 5.

[30] X. Bai, W., Zhang, G., Hsu, and Li-Ta, "Real-time GNSS NLOS detection and correction aided by sky-pointing camera and 3D LiDAR," presented at the *Proceedings of ION Pacific PNT 2019*, Honolulu, HA, USA, 2019.

[31] X. Bai, W. Wen, and L.-T. Hsu, "Using Sky-pointing fish-eye camera and LiDAR to aid GNSS single-point positioning in urban canyons," I*ET Intelligent Transport Systems*, vol. 14, no. 8, pp. 908-914, 2020.



Next-Generation HDR Vivid Video Technology Standard for Visual Arts

Quanhe Yu [1], Weiwei Xu [1], Yichuan Wang [1], Jiwu Zhang [1], Hu Chen [2], Le Yuan [3]
[1] Central Media Technology Institute
[2] Audiovisual Technology Laboratory, Munich Research Center
[3] Source Technology Development Dept, HiSilicon (Shanghai)

Abstract

Different display devices, such as televisions, computers, mobile phones, and tablets, have different display capabilities. The same content is displayed differently on all these devices. Usually, the image quality on these devices is not as good as the image quality achieved during production. Consumers cannot fully experience the artistic intentions of creators. To alleviate this problem, content creators must compromise on video quality during content production. Some industry standards have been formulated to ensure the display effect on different display devices. However, the display effect perceived by consumers is still not up to the standard required by creators. This paper proposes high dynamic range (HDR) Vivid video technology to analyze content characteristics during production, generate metadata, and adapt the luminance, contrast, and color on display devices based on the metadata, characteristics of the human visual system, and display capabilities of the devices, in order to faithfully reproduce artistic intentions. HDR Vivid optimizes various characteristics, including luminance, contrast, and color, and accurately tunes colors to improve visual effects and fully reproduce artistic intentions on different consumer devices, bridging the gap between creators and consumers.

Keywords

HDR, tone mapping, visual perception, luminance and contrast preservation, quality control, human eye perception model, optimization

1 Introduction

High-quality images depend on five elements: resolution, bit depth, frame rate, color gamut, and dynamic range.

Resolution is the number of pixels in a digital image [1]. For a given display size, a higher resolution indicates more pixels and finer details. According to BT.709 [2] released by the International Telecommunication Union (ITU), the resolution of high definition (HD) images is 1920 × 1080. In 2012, the ITU released the BT.2020 [3] to define 4K (3840 × 2160) and 8K (7680 × 4320) resolutions.

Bit depth is the number of color bits that can be displayed per pixel [4]. A larger bit depth indicates that more colors can be displayed, producing a more natural gradient of the entire image. BT.709 defines 8-bit encoding for HD images. BT.2020 increases the bit depth to 10 bits and 12 bits to meet the requirements of 4K and 8K images.

Frame rate is the number of images that are continuously displayed in a unit time (1s) [5]. The frame rate of a movie is usually 24p (24 frames per second). According to BT.709, the maximum frame rate of HD images is 60p. According to BT.2020, the maximum frame rate of 8K TV programs is 120p. At this frame rate, the smoothness of moving images is almost the same as that perceived in the real world.

Color gamut describes a range of colors that can be displayed [6]. The colors in the visible spectrum in nature form the largest color gamut, including all the colors that can be seen by human eyes. A larger color gamut indicates that more colors can be reproduced on a display device. BT.2020 extends the color gamut from Rec.709 (in BT.709) to Rec.2020, further expanding the color range.

Dynamic range is the difference between the maximum and minimum values of an object, often in the format of ratio [7]. The dynamic range of an image indicates the difference in luminance. A larger dynamic range indicates more bright and dark details in the image, which delivers a more vivid experience to consumers. BT.2020 significantly promoted the development of these elements, except for dynamic range. It was not until 2016 that ITU released BT.2100 [8] to officially define the parameters for HDR images.

The development of HDR technology requires the advanced capabilities of display devices. The UHD Alliance proposed the UltraHD Premium standard [9] for devices (such as televisions). Take luminance of a display device as an example. To pass HDR certification, the maximum luminance

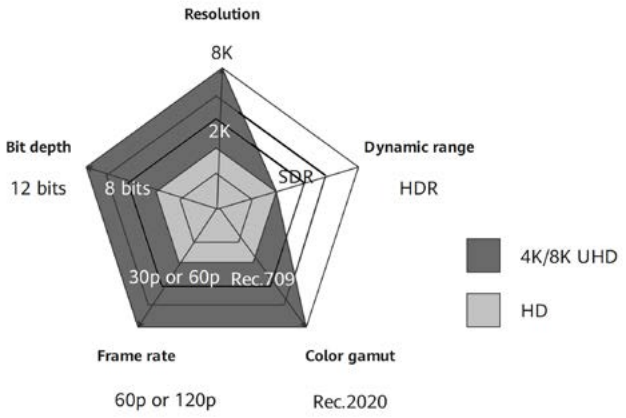


Figure 1 Elements of high-quality images

needs to reach 1000 nits (cd/m^2), and the maximum blackness (lowest luminance that can be displayed on the screen when it is not turned off) needs to be less than 0.05 nits (cd/m^2); alternatively, the maximum luminance needs to reach 540 nits (cd/m^2), and the maximum blackness needs to be less than 0.0005 nits (cd/m^2). However, high-quality HDR images produced by creators often have a larger dynamic range, which creates a daunting challenge: how to correctly display these images on devices with a lower dynamic range to fully reproduce the artistic intentions of creators. Tone mapping is a process of reducing the dynamic range while maintaining the contrast and details of the original image [10].

Research on tone mapping began years ago. Reinhard [11] developed a global tone mapping operator based on the response of cone cells in human eyes to light. Kim [12] proposed a scenario-specific mean logarithm luminance mapping model that follows Gaussian distribution based on the sensitivity of human vision. With the development of artificial intelligence, deep learning technology is also applied to the research on tone mapping. Patel [13] used generative adversarial networks to perform supervised learning on tone mapping. Zhang [14] used a multi-scale convolutional neural network to retain the maximum possible global and local information of an HDR image during mapping.

To further improve the display quality of display devices, the concept of metadata is used in engineering practices and standardization. Metadata includes the characteristics of the original image and is transmitted to devices. The display devices use the metadata to perform tone mapping based on the corresponding standards and specifications. In 2014, the Society of Motion Picture and Television Engineers (SMPTE) issued the ST 2086 standard [15], which defines

static metadata for HDR. Static metadata includes the information about the monitoring device used by creators during color tuning.

The proposed HDR Vivid technology has the following innovative features:

1. Uses a high resolution, large bit depth, high frame rate, wide color gamut, and large dynamic range to produce high-quality HDR Vivid content.
2. Provides a luminance, contrast, and color preservation method to ensure the flexibility, stability, and suitability of HDR content, and accurately reproduce the luminance, contrast, and colors in various scenarios.
3. Uses dynamic metadata based on the characteristics of human eyes to ensure consistent display effect of HDR content on devices with different display capabilities.
4. Uses content generation and production tools to expose the interface for adjusting dynamic metadata, enabling creators to freely create content.

HDR Vivid has been adopted by the China UHD Video Industry Alliance and has become an HDR video standard. This paper describes the principles and technical solution of HDR Vivid.

2 E2E HDR System

A creator often uses modern technology to create content. The creation process includes production, distribution, display, and perception.

Figure 2 shows the entire HDR creation process. First, a creator uses tools, such as a camera and color tuning software, to produce a master. Then, the master is compressed to a bitstream and transmitted during distribution. The compressed bitstream is decoded and presented during display and perception. These steps are interrelated, and the result of each step affects the next step. The following sections describe visual arts, as well as HDR production, distribution, display, and perception.

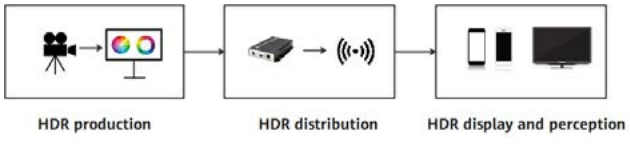


Figure 2 E2E HDR system

2.1 Visual Arts and HDR Production

Visual arts are the visual works created using certain materials and techniques. There are various forms of visual arts, including sculpture, painting, and photography, as well as various modeling techniques [16]. Visual arts are a "language" for conveying information. They are specifications or a symbolic system that uses basic visual elements and design principles to convey connotations [17]. Basic visual elements include line, shape, light, shade, color, texture, and space. They are used by artists to express artistic intentions. Creators select the materials and art forms (such as sculpture, painting, representational method, or abstract method), and use certain principles and methods to control the relationship between these elements within a certain scope, in order to develop an image with specific information [18].

Painting is an ancient form of visual arts [19]. Its development was greatly limited by technology. Cavemen used soil pigments to make cave murals. Bright dark blue pigments were extracted from celestine during the Renaissance to paint the Virgin Mary in elegant blue. The development of new pigments and the invention of the metal tube in the 19th century increased the possibility and convenience of outdoor painting. Camera obscuras helped painters achieve more accurate perspectives [20, 21] and led to the emergence of photography in the 19th century.

Photography was born with Gedar's daguerreotype. In 1889, Eastman Kodak Corporation of the United States used nitrocellulose film as the base material to produce photographic films on a large scale, which promoted the miniaturization of cameras. Photosensitive films use photosensitive silver salt particles. The optical characteristics of silver salt particles follow the gamma curve [22]. Intensifiers and reducers needed to be added on films with narrow spectral characteristics to produce good photos [23]. At the end of the 20th century, the invention of digital camera further promoted the development of photography. Tests conducted by photographers [24] found that the acceptable tolerance of film cameras and digital cameras is 8 and 9 respectively. In most cases, noises on films are difficult to distinguish from details. Digital cameras have a better performance.

From cave murals to sculptures, paintings, and electronic display devices, technological development has profoundly changed the way we create and perceive visual arts. Cathode Ray Tubes (CRTs) were widely used in electronic

display devices. The color gamut of P22 phosphor used by CRTs is close to the color gamut defined by BT.709, and the maximum luminance is close to 100 nits. Liquid crystal displays (LCDs) evolved rapidly to replace CRTs, providing a wider color gamut and higher luminance [25]. In recent years, organic light-emitting diodes (OLEDs), micro-LEDs, and laser display extended the color gamut to Digital Cinema Initiative-P3 (DCI-P3) or even the color gamut defined by BT.2020, the peak luminance to 3000–10,000 nits, and the bit depth to 10 bits. The development of HDR technology has improved visual perception in the same way that the development of mineral pigment purification improved to painting. HDR Vivid supports a depth of 12 bits, dynamic range of 4000–10,000 nits, BT.2020 color gamut (which includes the range of almost all colors that can be perceived by the human visual system), and metadata control, opening up more possibilities for art and enabling artists to present art more flexibly.

2.2 HDR Distribution

HDR content needs to be properly encoded for distribution. Bit depth and linear-to-nonlinear conversion are key factors for HDR coding. Digital images are discrete. The easiest way to retain more finer details is by increasing the bit depth. For a given maximum luminance, BT.2020 uses 10- or 12-bit coding to record more information than the 8-bit coding in BT.709.

Linear coding is the most direct mode for original HDR content. This coding mode uses many color levels for bright regions and does not record enough information about dark regions. As a result, the processed content is not suitable for human eyes due to the non-linearity of the human visual system. In the field of video, the gamma input/output characteristic curve is not only used for gamma correction in CRTs but is also used for other scenarios. An opto-electronic transfer function (OETF) transfers the data collected by optical sensors to electronic signals. An electro-optical transfer function (EOTF) restores and displays the coded data on display devices. In general, the curve of any type of OETF and the corresponding EOTF curve can be considered gamma curves. Research has been conducted on designing gamma curves for the human visual system, yielding promising results.

In the 18th century, Bouguer et al. proposed the log-uniform distribution, where the luminance perceived by human eyes increases uniformly when the logarithm of luminance increases uniformly. The Weber-Fechner Law [26] has been widely accepted. This model can be simply expressed as follows:

$$\frac{\Delta L}{L} = k \tag{1}$$

where L indicates the absolute luminance, and ΔL indicates the lowest luminance difference that human eyes can perceive at luminance L. k is a Weber fraction, which is a fixed value and can be solved by:

$$L(p) = Ce^{kp} \tag{2}$$

where p indicates the subjectively perceived luminance, and C is a constant. This model follows the log-uniform distribution. Log coding, which is commonly used, was developed based on the Weber-Fechner Law. The log-uniform model is precise in most cases. However, in a dark scenario (for example, where the luminance is lower than 1 nit [cd/m^2]) or a bright scenario (for example, where the luminance is higher than 1000 nits [cd/m^2]), the exponential-uniform model has a better performance. This concept was proposed by Plateau et al. in the 19th century. In 1957, Hunt [27] used optic nerve stimulation to prove that its relationship with light follows a power function. According to Stevens' Power Law proposed by Stevens [28], the luminance perceived by human eyes uniformly increases with the cubic root of luminance in a wide range. The widely accepted gamma coding was developed based on the Stevens' Power Law, but it uses different exponents. For example, in darkness, where rod cells of the retina that are more sensitive to light begin to work, the luminance perceived by human eyes approximately increases uniformly with the square root of luminance, according to the DeVries-Rose model [29].

As the dynamic range increases, the traditional method easily damages color levels or leads to a waste of a large number of code words. A new coding method based on the contrast threshold model was proposed. Two modes were designed for this method: (1) precisely design the code to adapt the contrast to human eyes; (2) approximately design the code to improve compatibility. The first mode is based on the Barten's Mode [30], which is a luminance perception model, including the pupil size, particle noise, lateral suppression of retinal cells, optic nerve noise, and impact of time and space on visual perception. The following contrast sensitivity function (CSF) is used to describe the result:

$$S(X_0,u,L) = \frac{1}{m_t} = \frac{\frac{M_{opt}(u,L)}{k}}{\sqrt{\frac{2}{T}\left(\frac{1}{X_0^2}+\frac{1}{X_{max}^2}\right)\left(\frac{1}{\eta pE(L)}+\frac{\Phi_0}{1-e^{-\left(\frac{u}{u0}\right)^2}}\right)}} \tag{3}$$

where m_t indicates the reciprocal of the CSF, L indicates luminance (unit: nit), u indicates the spatial frequency of the contrast (unit: cycle/degree), and X_0 indicates the angle of view (AOV) (unit: degree). Other variables are defined in ITU BT.2446 [31]. BT.2446 sets X_0 to 60 degrees, turning the function into a binary function $S(u, L)$. This model was first used by the Digital Imaging and Communications in Medicine (DICOM) standard [32] in 2001 to develop a coding scheme for 0.05 to 4000 nits (cd/m^2) based on $u = 4$ cycles/degree. In 2008, Aydın [33] et al. proposed Perceptually Uniform (PU) coding. PU selects the spatial frequency point with the highest contrast sensitivity for each luminance and converts sensitivity S into a monodrome function of L. PU uses the lookup table (LUT), limiting the application scenarios. In 2014, SMPTE officially released the Perceptually Quantizer (PQ) coding function in the ST.2084 [34]. PQ uses a CSF with an AOV of 40 degrees, provides a luminance range from 0 to 10,000 nits, and can be used in coding schemes that have a bit depth of 10 to 12 bits. PQ uses a simplified and easy-to-inverse approximation function to replace the LUT in PU, significantly improving usability. In 2016, BT.2100 defined PQ as one of the two gamma curves recommended for HDR program distribution, and PQ has since become the mainstream format for HDR content distribution of Internet videos and movies.

However, in precise coding, the coded data must be mapped to an absolute luminance, instead of being mapped between 0 and 1 based on the highest luminance of a display in traditional coding. Without luminance adjustment, PQ cannot ensure the display effect of HDR content for all displays. This is because luminance-based decoding truncates the highlight information that exceeds the maximum display capability of the device. A simplified or "simplistic" coding mode hybrid-log gamma (HLG) [35] was developed by British Broadcasting Corporation (BBC) and Nippon Hoso Kyokai (NHK) to improve device compatibility and allow devices with different display capabilities to use the same signal source. HLG integrates the traditional visual perception model, which uses approximate square root coding for dark regions and logarithmic coding for bright regions. This method features good compatibility (with SDR). The HLG coding function based on relative luminance has been selected by BT.2100 as a recommended gamma curve for HDR program distribution, due to the fact that the maximum luminance of consumer displays is not very high. HDR is widely used in many fields, such as broadcast TV and live streaming.

2.3 HDR Display and Perception

Content is displayed based on the display capabilities and policies of display devices. The visual system, visual nerve, and brain vision forms a perception of the displayed content in the brain.

Human eyes, as an optical system, can be described by Fourier optics. A retina image can be represented as a convolution of the input image with a point spread function (PSF). However, human eyes are not a perfect optical system. The PSF of human eyes is a fuzzy kernel in a wide AOV range, as shown in Figure 3. The PSF depends on the pupil aperture, and the aberrations of the cornea and crystalline lens. The wide AOV range is caused by scattering. The center of the PSF causes blurs. At the tail, light sources of bright spots are superposed on other pixels, reducing the contrast of the image. In addition, the dependence of refractive index on wavelength causes chromatic aberration. Light with long and short wavelengths form images at different depths on the retina. Blue is usually more blurred than other colors. An eye is equivalent to a band-pass filter for light reaching the retina. It adapts the sensitivity of the photoreceptor to the wavelength and absorbs more blue light. The visual system also uses the neural network to compensate the aberrations. Although the optical quality of human eyes is low, they are still a highly optimized optical system [36].

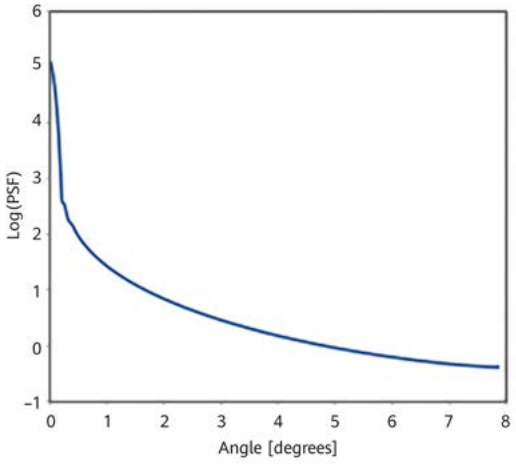


Figure 3 PSF of the visual system [36]

The retina is 0.25 mm thick and has five layers. The outer nuclear layer contains photoreceptor cells. The outer plexiform layer contains the synaptic terminals of photoreceptors and dendrites of bipolar and horizontal cells. The inner nuclear layer contains bipolar, horizontal and vertical cell bodies. The inner plexiform layer contains axonal

terminals of bipolar cells. The ganglion cell layer contains all retinal ganglion cells. The axons of retinal ganglion cells form the optic nerve, which exits the retina through the optic disk and connects to the brain [37]. Approximately 90% of the optic nerve fibers terminate in the lateral geniculate nucleus (LGN), which is a small part of the thalamus. Starting from the retina, the parallel pathways reach the primary visual cortex (V1) through the LGN.

The visual system can adapt to the content and viewing environment, including the luminance, light, and contrast. Light adaptation is completed on the retina. The effect is equivalent to dividing the luminance by average luminance. The retina can convert a wide range of visual signals into a small range of neural signals. Contrast adaptation starts on the retina and is enhanced in the LGN and visual cortex. The effect is equivalent to dividing the contrast at the corresponding pixel position by local contrast [38].

Figure 4 shows the response of the nervous system, which is usually an S curve of the Naka-Rushton equation:

$$R = R_{max}\frac{I^n}{I^n + I_s^n} \tag{4}$$

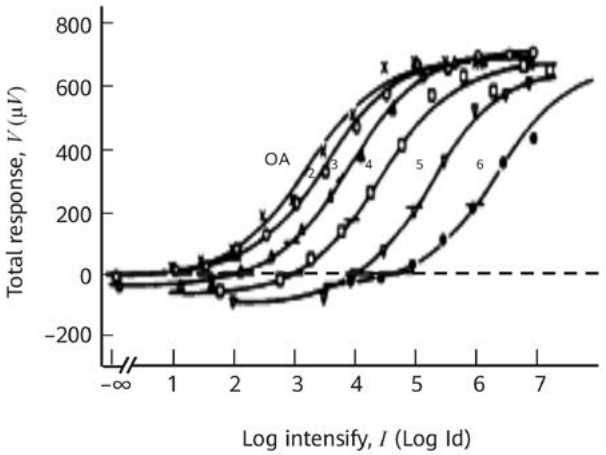
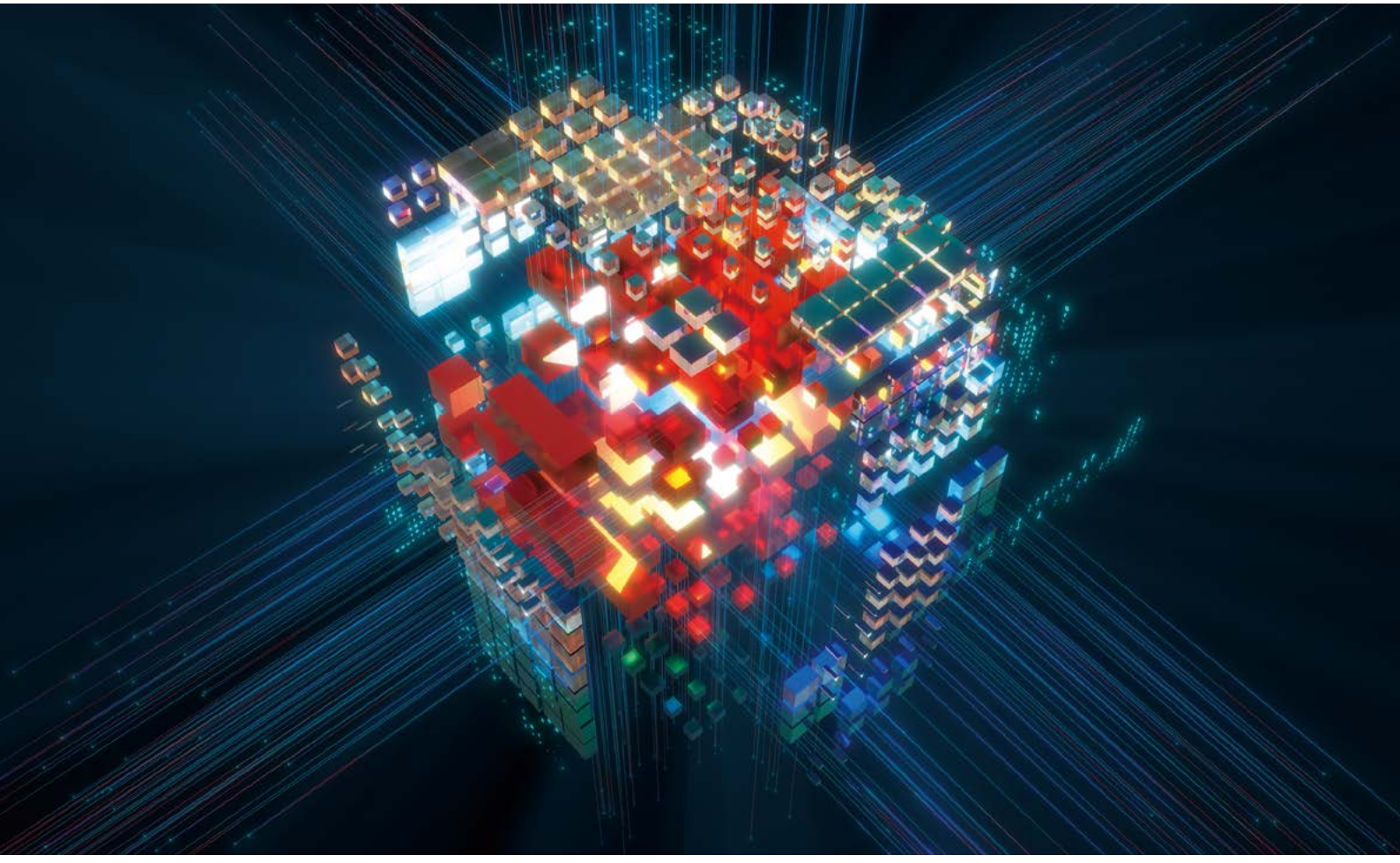


Figure 4 Photoreceptor response curve [39]

where R indicates the response to input signal I, R_{max} indicates the maximum response, I_s indicates a half-saturation coefficient, and n is a fixed exponent [38]. n is usually 0.75 for cones, 1.0 for retina, 1.1 for LGN, and 2 for V1. In general, a small input leads to an approximately linear curve, a medium input leads to an exponential curve, and a large input leads to a logarithm curve.



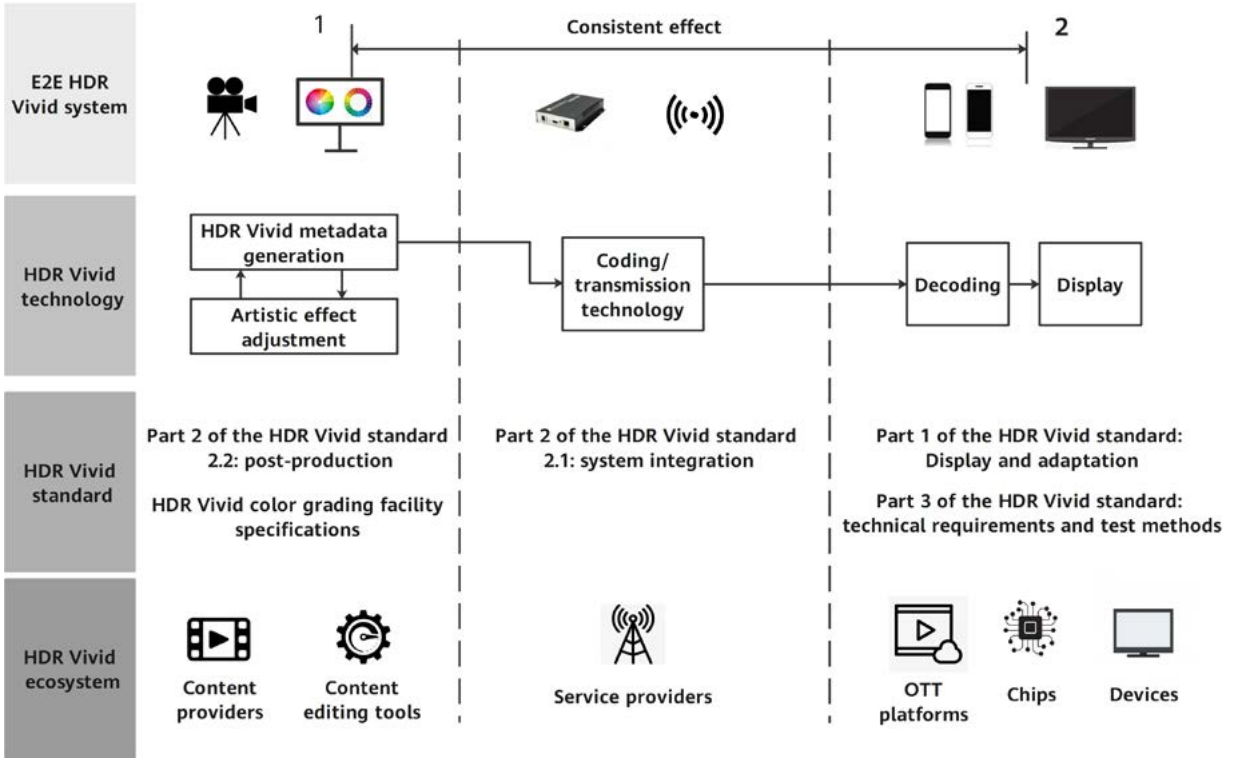


Figure 5 E2E HDR Vivid technology enabling visual arts

3 HDR Vivid Technology

3.1 E2E HDR Vivid System

Conventional static HDR only allows video creators to control the image quality on the device in the production environment. To guarantee the display effect on display devices, creators have to manually debug different video versions for different viewing environments on all difference devices. This is time-consuming and infeasible. The HDR Vivid technology helps creators faithfully reproduce artistic intentions on different display devices. Figure 5 shows the E2E system block diagram of the HDR Vivid technology. Each production process involves the HDR Vivid technology, standard, and ecosystem. The HDR Vivid technology ensures that the artistic intentions of creators can be reproduced on different display devices. The HDR Vivid standard ensures interoperability. The HDR Vivid ecosystem facilitates the implementation and provides support for the HDR Vivid technology and standard.

To reproduce the artistic intentions of creators, results of 1 and 2 in Figure 5 need to be as similar as possible. The mathematical model is shown in Figure 6 and described by Formula (5).

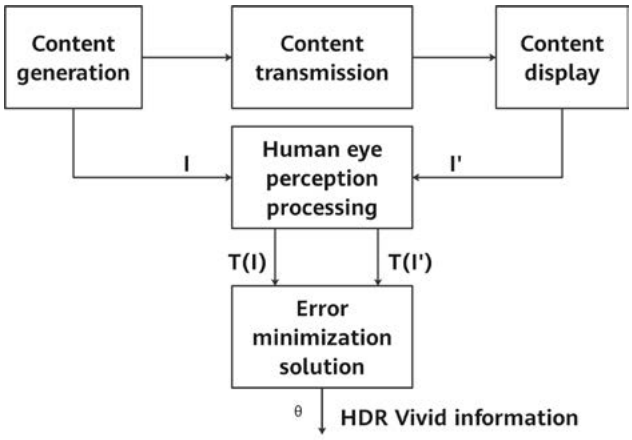


Figure 6 Minimization solution of the human eye perception system

$$\underset{\theta}{\operatorname{argmin}} E\left(\left\|\mathrm{T}(\mathrm{I})-\mathrm{T}(\tilde{I})\right\|_2^2\right) \tag{5}$$

Formula (5) is a minimization solution problem. Tremendous efforts have been made by researchers to solve this problem. Z. Mai [40] assumed that only luminance is lost and chroma is not lost in $\tilde{I}$, and proposed a method for processing luminance loss. This method divides the input luminance into equal parts, performs differentiated processing on each part, and uses the Karush–Kuhn–Tucker optimization theory to calculate the optimal analytical

solution. In addition, various models were proposed based on different visual experiments, including the CSF model [41], CAM model [42], Hunt model [43], contrast invariant visual distortion model (CIVDM) model [44], and other human eye perception models [45–47]. Human eye perception models are developed based on the human eye perception system and simulate the responses of human brains to images. Various factors, including the responses of human eyes to color and luminance stimulation, luminance of display devices, and luminance and colors of videos, need to be considered to improve these models.

HDR Vivid uses all the factors that have a great impact on the display effect in human eye perception models to design the HDR production and presentation processes. The HDR Vivid technology analyzes the content based on human eye perception models and generates HDR Vivid metadata, which includes luminance and color curves. After HDR content distribution, the metadata is parsed and used in the display process to reproduce the artistic intentions of creators based on the display capability of the device. HDR Vivid information includes the light, shadow, and tone information of each image. This information is used throughout the entire E2E HDR Vivid system, including presentation and production technology.

3.2 HDR Vivid Presentation Technology

3.2.1 Tone Mapping for Luminance and Contrast Preservation

The display capabilities of display devices are usually different from the dynamic range of the original content. Tone mapping needs to be performed to adapt the dynamic range of the content to the display capabilities of the devices. During tone mapping, the luminance and contrast of the content need to be as close as possible to the original luminance and contrast.

Previous studies show that the response of the visual nerve system varies with the maximum value, average value, and variance of the stimulus. The response of the visual nerve system is usually described by the S curve of the Naka-Rushton equation [48]:

$$R = R_{max} \frac{L^n}{L^n + Ls} \tag{6}$$

where n is usually 0.75 for conical cells, 1.0 for retina, 1.1 for LGN, and 2 for V1. In general, a small input luminance leads to an approximately linear curve, a medium input luminance leads to an exponential curve, and a large input luminance leads to a logarithm curve [49]. In different test environments, the equation can be [50]:

$$R = R_{max} \frac{(g(L-c))^n}{(g(L-c))^n + 1} \tag{7}$$

where g, n, and c vary according to the test content, and L is the log value of the luminance. The Steven model uses the following luminance perception function in the form of a cubic root to best describe how human eyes perceive highlights:

$$R = c\,(a * L + b)^{0.33} + d \tag{8}$$

where L is the input, R is the output, and a, b, c, and d are parameters.

The perception of the original luminance L_{org} and the luminance after tone mapping L_{TM} needs to be the same to ensure consistent luminance perception.

$$R\,(L_{org}, MaxL_{org}, avgL_{org}) = R\,(L_{TM}, MaxL_{TM}, avgL_{TM}) \tag{9}$$

where $MaxL$ and $avgL$ are the maximum and average luminance of the image respectively.

Research shows that changes in contrast perception are affected by the spatial frequency of the stimulus value [51]. Generally, kernels of Fourier transform are used to simulate convolution kernels in the visual system to study the CSF. The contrast perception sensitivity is a reciprocal of the contrast perception threshold, and the contrast perception threshold indicates the minimum contrast that can be perceived on a uniform background. Barten used luminance, AOV, background, pupil size, photoreceptor sensitivity, and other factors to develop a fine CSF model [52]. This model has a maximum contrast sensitivity value for each luminance value. A CSF is used in DICOM, PU, and PQ to draw the photoelectric and electro-optic transfer curves with a uniform contrast. In these curves, the quantization step between different gray values is less than the Just Noticeable Difference (JND) of human eyes. To ensure uniform contrast perception, the PQ curve of HDR Vivid should be nearly linear around the average value, and the luminance curve of the main body should also be nearly linear (a long and narrow curve of the Naka-Rushton equation can meet the requirements). The basic HDR Vivid curve can be derived based on luminance perception and contrast perception:

$$m\_a \times \left(\frac{m_p \times PQ(L)^{m_n}}{(K1 \times m_p - K2) \times L^{m_n} + K3} \right)^{m\_m} + m\_b \tag{10}$$

Research shows that the overall response of the visual nerve system varies depending on the maximum value, average value, and variance of the stimulus. According to the Steven model, luminance perception in bright regions follows a uniform distribution of cubic roots. According to the DeVries-Rose model, luminance perception in dark regions follows a uniform distribution of square roots, which is almost the same as the approximate result of the response of the Naka-Rushton equation at different positions. In the Naka-Rushton equation, a small input luminance leads to a linear curve, a medium input luminance leads to an exponential curve, a high input luminance leads to a logarithm curve. Research [53, 54] found that the Naka-Rushton equation alone cannot best describe the luminance perception in this case, as shown in Figure 7.

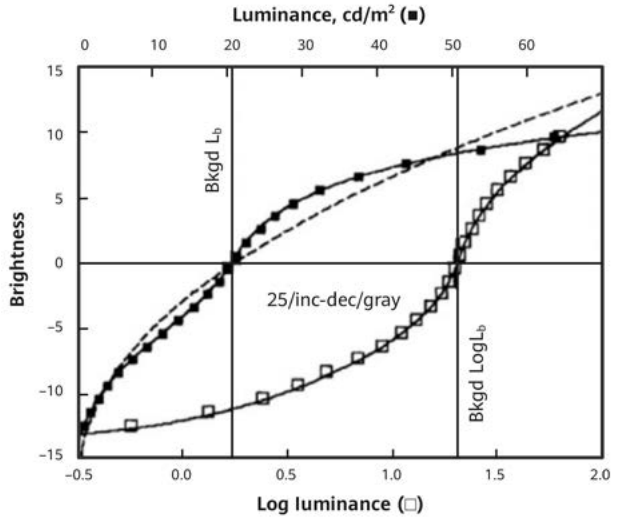


Figure 7 Response of human eyes to changes of average luminance [53]

In this figure, the response still meets the Naka-Rushton equation around the average luminance. However, the responses in bright and dark regions are different. This can be explained by the structural characteristics of human eyes. The retina consists of two types of photosensitive cells: cone cells and rod cells. These cells have different responses in bright and dark regions [55]. Only rod cells respond between 10^{-6} and 10^{-2} nits. Cone cells respond more actively above 10 nits. For medium luminance, the response of cone cells increases with luminance.

Therefore, the HDR Vivid standard uses a spline function to process the tone mapping result of dark regions and describe the response in dark regions:

$$a_{n+1}(x-T_n)^3 + b_{n+1}(x-T_n)^2 + c_{n+1}(x-T_n)^1 + d_{n+1} \quad T_n \le x < T_{n+1} \tag{11}$$

where x is the input luminance, and a, b, c, and d are spline parameters.

The HDR Vivid tone mapping curve can be described by Equation (12). Figure 8 shows the curve.

$$Phoenix = \begin{cases} a_1 x + b_1 & 0 \le x < T_1 \\ a_{n+1}(x-T_n)^3 + b_{n+1}(x-T_n)^2 + c_{n+1}(x-T_n)^1 + d_{n+1} & T_n \le x < T_{n+1} \\ m\_a \times \left(\dfrac{m\_p \times L^{m\_n}}{(K1 \times m\_p - K2) \times L^{m\_n} + K3} \right)^{m\_m} + m\_b & T_k \le x < T_{k+1} \end{cases} \tag{12}$$

This function has three parts. Part 1 is a primary spline curve, which accurately describes the dynamic range of dark regions. Part 2 is a cubic spline curve, which uses HDR characteristics to generate the expected curve at any given position, ensuring that the curve is applicable to all HDR scenarios. Part 3 is a basic curve, which partly ensures the luminance and contrast of the main body. In addition, a few control parameters can be set to improve the display effect.

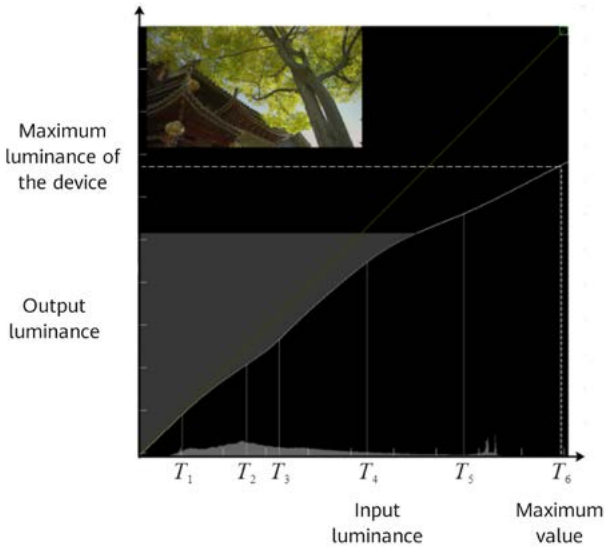


Figure 8 HDR Vivid luminance processing curve

3.2.2 Saturation Adjustment for Color Perception Preservation

HDR Vivid tone mapping introduces a luminance gain to ensure color accuracy. This gain is used to process R, G, and B components separately at the same time with the same ratio.

$$\begin{aligned} Rnew &= R * gain \\ Gnew &= G * gain \\ Bnew &= B * gain \end{aligned} \tag{13}$$

The conversion between XYZ and linear RGB is a linear conversion. Therefore:

$$\begin{aligned} Xnew &= X * gain \\ Ynew &= Y * gain \\ Znew &= Z * gain \end{aligned} \tag{14}$$

In CIE xy(x, y, Y):

$$\begin{aligned} xnew &= \frac{X * gain}{X * gain + Y * gain + Z * gain} = x \\ ynew &= \frac{Y * gain}{X * gain + Y * gain + Z * gain} = y \end{aligned} \tag{15}$$

These equations can be used to prevent a large color deviation during luminance mapping.

Chroma also changes with luminance. Tumblin and Turk's research [56] adjusted colors based on luminance changes:

$$Cnew = \left(\frac{C}{L}\right)^{s} * L_{TM} \tag{16}$$

This equation evolved in later research [57, 58]:

$$Cnew = \left(\frac{L_{TM}}{L}\right) * C \tag{17}$$

Our research found that the color appearance of an object changes obviously with the overall luminance due to the Hunt effect [59]. Chroma changes with luminance, which affects color perception. How much the color perception changes depends on the colors and luminance of the scenario. Therefore, the adjustment range needs to be controlled:

$$Vnew = \left(\frac{L_{TM}}{L}\right)^{C0} * V \tag{18}$$

$$Unew = \left(\frac{L_{TM}}{L}\right)^{C0} * U \tag{19}$$

This equation adequately compensates for the chroma changes caused by luminance changes.

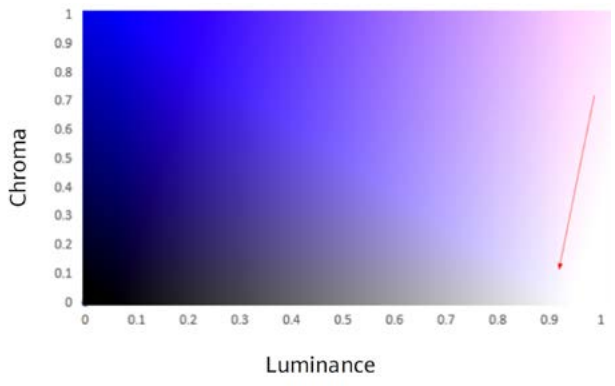


Figure 9 Luminance and chroma

In addition, the experiment showed that in contrast with medium bright regions, the slope of chroma changes more greatly with luminance in bright regions, as shown in Figure 9.

Special processing is required for bright regions:

$$S_{ca} = \begin{cases} B - C1 \times SatR \times \left(\dfrac{f_{MAX}[i] - A \times RML}{RML - A \times RML}\right)^{M} & TML < f_{MAX}[i] < RML \\ B - C1 \times SatR & f_{MAX}[i] \geq RML \end{cases} \tag{20}$$

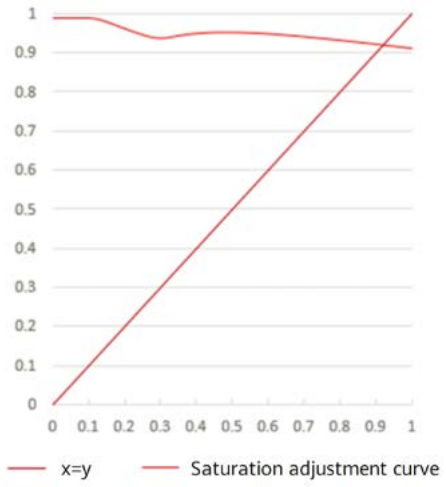


Figure 10 HDR Vivid color processing

3.3 HDR Vivid Production Technology

Figure 10 shows the overall saturation adjustment curve based on the luminance gain of tone mapping.

The HDR Vivid production technology is key to the entire process. This technology uses dynamic metadata transferred from end to end to ensure a consistent display effect on different display devices, and allows creators to customize the artistic style. Figure 11 shows the HDR Vivid production process.

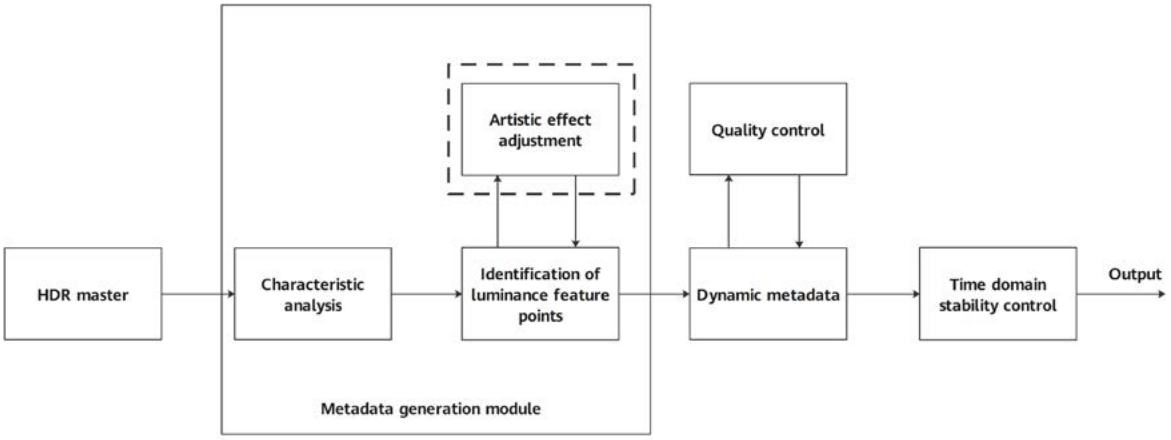


Figure 11 HDR Vivid production process

First, the basic features of the HDR master are analyzed, including the luminance and histogram. Then, various algorithms are used to determine the key control points (also called luminance feature points) of a mapping curve based on the features in different scenarios. Creators can adjust the luminance feature points. After the luminance feature points are determined, related information is converted into dynamic metadata according to the HDR Vivid standard. These processes are performed by the metadata generation module, as shown in Figure 11. The quality control module is designed to evaluate and provide feedback on the quality of dynamic metadata without human intervention. The time domain stability control module prevents flicker caused by dynamic metadata. The following sections describe the technologies used in these modules.

3.3.1 Metadata Generation

The maximum value, minimum value, average value, and variance (variation range) of luminance are the basic features of an image. Histogram is a two-dimensional statistical chart of data distribution. This concept was first proposed by British statistician Karl Pearson [60] and has been widely used in various fields. In a luminance histogram, x axis indicates the luminance of a pixel, and y axis indicates the number of pixels corresponding to the luminance. A luminance histogram reflects the luminance distributions of an image and is key to identifying luminance feature points.

In research on tone mapping and histogram equalization, an image is usually divided into many regions based on the histogram for differentiated processing, in order to reach a balance between contrast and luminance. [61] divides an image into bright and dark regions by estimating the luminance and reflectivity of different parts in the image. Given that illumination is usually uneven in nature, [62] further divides an image into underexposure (dark), normal exposure, and overexposure (bright) for differentiated processing. However, a lack of precision in the operations on the normal exposure region will cause contrast losses. [63] further divides the normal exposure region into low, medium, and high to improve the display effect. More precise division improves the image effect. However, most of the images discussed in previous research are too dark or too bright, which do not meet the current requirements. HDR Vivid uses the curve generation algorithm to generate metadata. This algorithm precisely divides an image or histogram based on the luminance and luminance distribution of different statistical characteristics of the content.

The luminance of a natural image follows Gaussian distribution. According to this principle, JPEG uses a transform kernel to transform the image into the transform domain for quantization and compression. Gauss distribution was first proposed by German mathematician and astronomer Moivre in 1733. German mathematician Gauss first applied it to research on astronomy. The normal curve of the distribution is in bell shape, as shown in Figure 12.

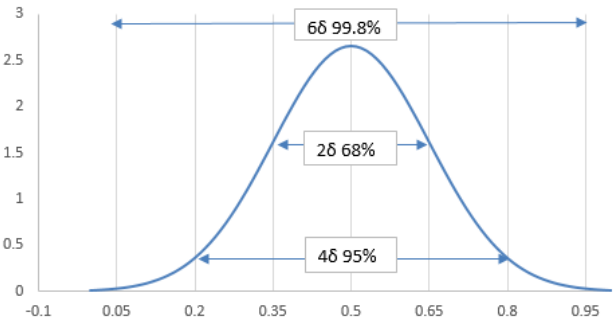


Figure 12 Gaussian distribution

The curve has six segments, accounting for 99.8% of the total pixel distribution. Based on the luminance distribution, HDR Vivid metadata divides an image into six segments, including extremely dark, dark, low bright, medium bright (skin tone), diffuse white, and highlight reflection. Extremely dark indicates the scotopic vision of human eyes, where most colors are lost, the noise is heavy, and absolute darkness occurs. Dark is brighter than extremely dark. In this region, colors are dark but visible, and texture is clear. In contrast, the extremely dark region does not have texture. Medium bright refers to the objects (such as skin tones) with reflectivity close to 18%. Diffuse white is the extremely bright region in SDR and the second bright region in HDR. Highlight reflection is the extremely high luminance caused by materials with a high reflectivity in nature. Figure 13 shows a typical HDR Vivid mapping curve, which consists of several smoothly connected curves. These curves are connected by the luminance feature points to be identified. The mapping curve has six luminance feature points. The shapes, design principles, and method for identifying the luminance feature points on each curve are described as follows.

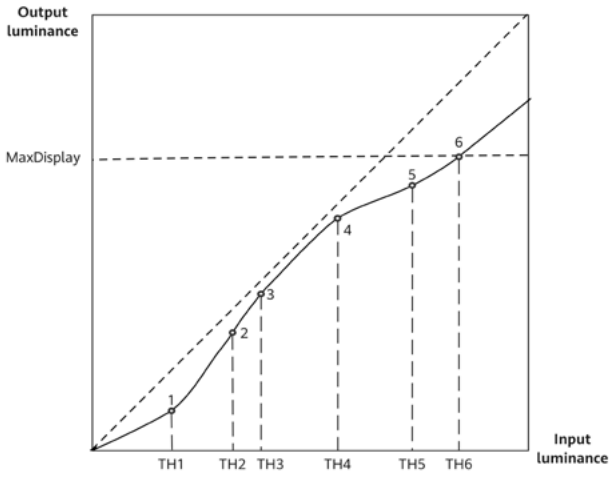


Figure 13 Typical HDR Vivid mapping curve

Based on research of Mantiuk et al. [44], the horizontal coordinate TH1 of luminance feature point 1 is set to 0.15, which is a PQ-encoded value. Unless otherwise specified, all luminance reference values in this section are PQ-encoded. The research shows that the lowest luminance that can be perceived by cone cells that are responsible for color perception is about 1 nit (0.15). If the luminance is lower than this value, rod cells that cannot perceive color work predominantly. In general, the start point of the curve is the origin (0, 0), and the start point is connected to luminance feature point 1 by the primary spline. The advantage of this design is that the shape of the curve can be controlled only by adjusting the slope. The slope can be dynamically adjusted based on the distribution of the pixels below TH1 in the luminance histogram. If only a few pixels are distributed in this area, the slope can be reduced to improve the contrast. If many pixels are distributed in this area, the slope needs to be increased to prevent detail loss caused by overcompression of dark regions. After a slope is selected, the vertical coordinate of luminance feature point 1 can be calculated.

Luminance feature points 3 and 4 are connected by a basic curve, which is described by:

$$F(L) = m\_a \times \left(\frac{m\_p \times L^{m\_n}}{(K1 \times m\_p - K2) \times L^{m\_n} + K3} \right)^{m\_m} + m\_b \quad (21)$$

This curve is flexible and determines the image style after mapping. The experiment shows that we can keep m_m, m_n, m_b, $K1$, $K2$, and $K3$ unchanged (for example, $m\_m = 2.4$, $m\_n = K1 = K2 = K3 = 1$, $m\_b = 0$) and adjust the values of m_p and m_a to achieve the optimal result. m_p depends on the overall luminance of the scenario. m_a depends on the maximum luminance of the input content and the maximum display capability of the display device. Therefore, the coordinates of (Maxsource, MaxDisplay) and (Avgsource, AvgLight) can be used to solve m_p and m_a. Maxsource is the maximum luminance in the image characteristics, MaxDisplay is the maximum display capability of the display device, and Avgsource is the average luminance in the image characteristics. AvgLight is calculated as follows:

$$AvgLight = \frac{\sum_{i=0}^{N_{frame}-1} q(i)}{N_{frame}} \quad (22)$$

$$q(i) = \begin{cases} MaxDisplay & L[i] \geq MaxDisplay \\ L[i] & Others \end{cases} \quad (23)$$

where N_{frame} indicates the total number of pixels in a frame of the image, and $L[i]$ is the luminance of pixel index i. It should be noted that based on the research of Mantiuk [44], using (Perceptual_1nit, 0.15) to replace (Avgsource, AvgLight) when Avgsource is small better matches the visual characteristics of human eyes. The preset lower threshold of Avgsource is usually 0.25. In addition, the coordinates of two points that can maximize the local contrast can be solved by dividing several luminance intervals and using histogram equalization [64] in order to calculate m_p and m_a. Perceptual_1nit can be calculated by:

$$N(Perceptual\_1nit) = (N(0.25) - N(0.15)) \times Rate + N(0.15) \quad (24)$$

where $N(x)$ indicates the number of pixels with a PQ-encoded value less than x in the luminance histogram, and $Rate$ is the preset ratio, which is usually 30%.

According to ITU BT.2408 [65], the lower limit of the luminance range of skin tones is 0.35. Therefore, the horizontal coordinate TH3 of luminance feature point 3 is set to 0.35 to ensure the display effect of skin tones. The vertical coordinate of luminance feature point 3 can be solved using TH3 in the basic curve function. The horizontal coordinate of luminance feature point 2 depends on the pixel distribution in [TH1, TH3] and can be calculated by equations (25) and (26):

$$TH2 = \frac{\sum_{i=0}^{N_{frame}-1} q(i)}{Num} \tag{25}$$

$$q(i) = \begin{cases} L[i] & TH1 \le L[i] \le TH3 \\ 0 & Others \end{cases} \tag{26}$$

The definitions of N_{frame} and $L[i]$ are the same as above. Num indicates the number of pixels in [TH1, TH3] in the luminance histogram. The initial luminance feature point 2 is on a straight line connecting luminance feature points 1 and 3. The vertical coordinate of luminance feature point 2 can be calculated using the known horizontal coordinate of luminance feature point 2. To improve the contrast, an offset can be added to the vertical coordinate of luminance feature point 2 based on the relationship between $Num1$ (the number of pixels in [TH1, TH2]) and $Num2$ (the number of pixesl in [TH2, TH3]). To improve flexibility and protect the details in dark regions, two cubic spline curves are selected to smoothly connect luminance feature points 1 and 3. The cubic spline curves can be described by equation (27):

$$F(L) = a_{n+1}(L-T_n)^3 + b_{n+1}(L-T_n)^2 + c_{n+1}(L-T_n)^1 + d_{n+1}$$
$$T_n \le L < T_{n+1} \tag{27}$$

The horizontal coordinate TH6 of luminance feature point 6 is generally set to the maximum luminance Maxsource in the image characteristics. The vertical coordinate is set to the maximum display capability MaxDisplay of the device to fully leverage the display capability of the device and establish a mapping relationship between the maximum luminance of the image and the maximum luminance of the display. The horizontal coordinate TH6 of luminance feature point 6 can be adjusted based on the distribution around Maxsource in the luminance histogram. To calculate the horizontal coordinate of luminance feature point 4, we can evenly divide [TH3, TH6] into U intervals. The preset value of U is 6. The initial value of TH4 can be calculated using equation (28):

$$TH4 = TH3 + \left(\frac{TH6 - TH3}{U}\right) \times (U-2) \tag{28}$$

Then, the updated TH4 and the horizontal coordinate TH5 of luminance feature point 5 are solved using histogram equalization in [TH4, TH6]. Luminance feature point 4 is also on the basic curve. The vertical coordinate of luminance feature point 4 can be solved using TH4 in the basic curve function. Similar to luminance feature point 2, the initial vertical coordinate of luminance feature point 5 can be solved using TH5 in the straight-line equation for luminance feature points 4 and 6. An offset can also be added to luminance feature point 5.

The calculation result of the luminance feature points and related parameters in the solution process are converted based on the standard form and requirements of HDR Vivid dynamic metadata in order to generate dynamic metadata.

The artistic effect adjustment module depends on HDR content production tools. HDR Vivid uses a black box to dynamically adjust metadata to adapt to creators' habits and simplify tool usage. Common artistic effect adjusting elements are designed for creators on the tool GUI, such as dark region details, medium gray luminance, and highlight. Creators can update the dynamic metadata by simply dragging and dropping these elements. The updates will be mapped to the corresponding luminance feature points. Figure 14 shows a simplified GUI.

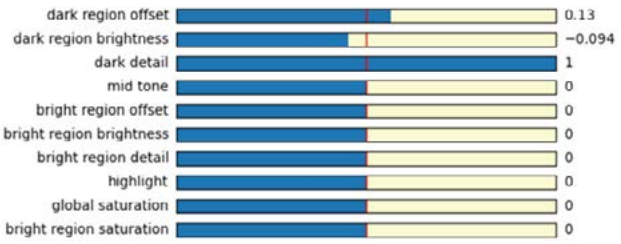


Figure 14 Example of an HDR Vivid artistic effect adjustment GUI

3.3.2 Quality Control

Figure 15 shows the HDR Vivid quality control system. This is an intelligent control system with adaptive adjustment, which can export optimal HDR Vivid metadata. In addition, parameter settings can be manually adjusted to produce special artistic effects.

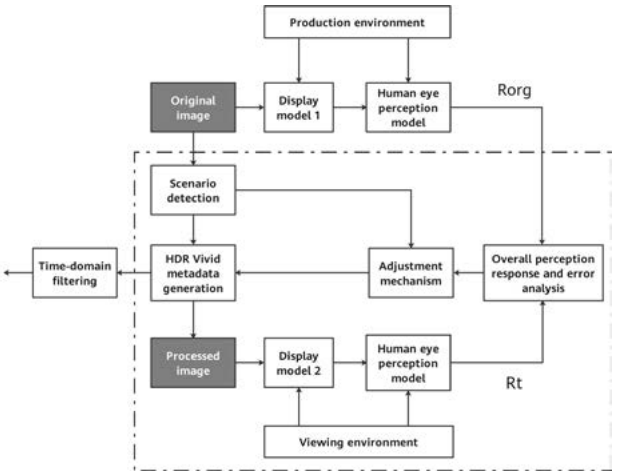


Figure 15 HDR Vivid quality control system

The display model in Figure 15 can be described by equation (29) [66]:

$$L_d(L') = OETF^{-1}(L') \cdot (L_{max} - L_{black}) + L_{black} + L_{refl} \quad (29)$$

where L_d indicates the luminance of the display, and L' indicates the input luminance, which is usually the electrical signal obtained by decoding the code stream. $OETF^{-1}(L')$ converts L' from electronic signals to optical signals. Other common transfer functions are gamma, HLG, and PQ. L_{max} is the peak luminance of the display. L_{black} is the black level of the display. L_{refl} is the luminance of ambient light reflected on the screen, which can be approximated by the equation (30):

$$L_{refl} = \frac{k}{\pi} E_{amb} \quad (30)$$

E_{amb} is the luminance of ambient light. k is the reflectivity of the display.

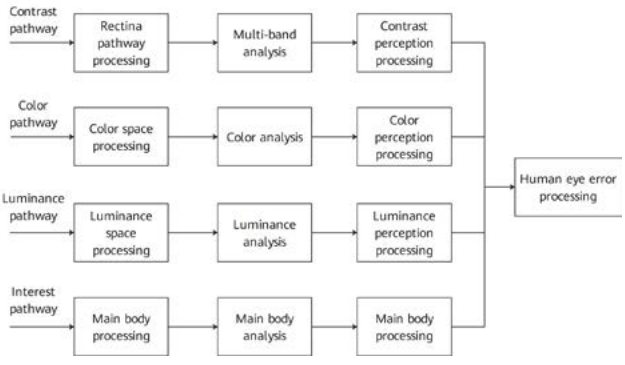


Figure 16 Flowchart of the HDR Vivid human eye perception pathways

Figure 16 shows how human eye perception models simulate the reaction of human eyes to a video. These models extend the CIVDM contrast model by adding color, luminance, and interest pathways. They fully consider the loss of HDR Vivid content during transmission and display, and the characteristics of HDR Vivid content. The color pathway maintains the wide color gamut of HDR, especially highly saturated colors. The luminance pathway reproduces HDR highlight details. The interest pathway ensures that key HDR Vivid content is not lost. The human eye error processing module evaluates the results of the contrast, color, luminance, and interest pathways to select the most important part to reproduce artistic intentions to the greatest extent.

In the contrast pathway, the retina pathway considers various factors, such as the screen size and distance, and converts the input signals into LMSR signals that can be more easily perceived by human eyes. During multi-band analysis, the LMSR signals are decomposed into multiple frequency band signals by Laplacian frequency band decomposition. Then, the contrast perception model uses a CSF to process the signals on each frequency band, and then process all the CSFs together. In the color pathway, color space processing decomposes signals into lab and XYZ signals. During color analysis, the color components ab and xy in the lab and XYZ signals are used to calculate the color saturation, tone, and position in a horseshoe color gamut diagram. During color perception processing, the color offset, position, and correction direction are calculated based on the color analysis result. In the luminance pathway, luminance space processing converts signals into lab and YCbCr signals, and uses the guilder filter to classify Y signals into the detail layer and the luminance layer. During luminance analysis, highlight details are extracted from the luminance layer and form the l signals of lab signals. Highlights that do not contain details are discarded. During luminance perception processing, the luminance layer and l layer are used to calculate how many highlight details are lost. In the interest pathway, the image edges are extracted and processed during main body processing, and the image scenario (people, plants, animals, or others) is detected when the main body is analyzed. Based on the edge detection result, the area of interest is identified from the main body, and the loss of information in this area is calculated when the main body is processed.

When human eye errors are processed, a non-linear combination of the contrast perception response, color perception response, luminance perception response, and area of interest perception response exported by the last module is used to calculate the overall perception error. For example, for an image with a high contrast, the model focuses on whether the image loses contrast after processing. For an image with many highlights and highlight details, the model focuses on whether the luminance details

are lost after processing. For an image with highly saturated colors, the model focuses on whether the color deviation is large after processing.

The quality control system makes adjustments based on the image loss. If the contrast loss in a region is large, the slope of the contrast curve for this region will be increased to improve the contrast. Analysis shows that many pixels are distributed in the medium bright area. If the contrast in this area is reduced during processing, the response parameters need to be adjusted to maintain the original contrast in this area. The luminance contrast of the main body is significantly improved, as shown in Figure 17.

If a large number of highlight details are lost, highlight details will be enhanced. As shown in Figure 18, highlight details in a snow image are significantly improved.

If the saturation loss is large, the color saturation factor is adjusted to maintain the original colors, as shown in Figure 19.

The manual adjustment model is used to manually perform adjustments on the result. It can be used to change the overall perception response and error analysis module to adjust the loss of a specific characteristic, for example, increasing the contrast loss. In addition, this model can change the adjustment mechanism to meet the requirements of artists and colorists.

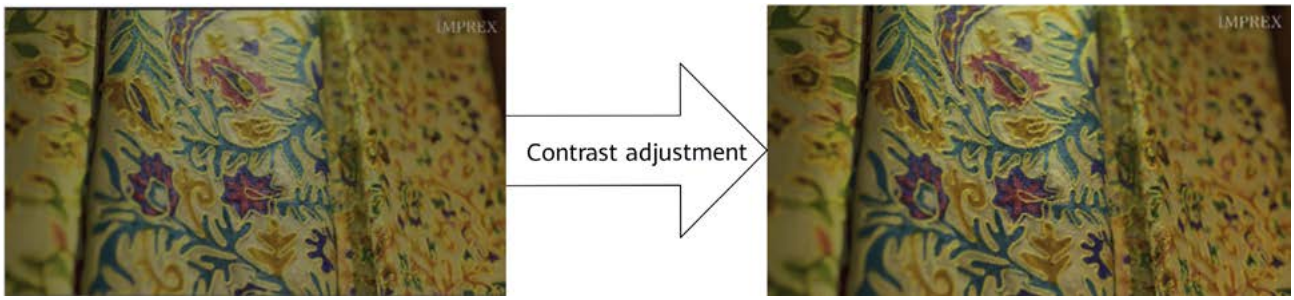


Figure 17 HDR Vivid main body contrast adjustment

Figure 18 HDR Vivid highlight details adjustment

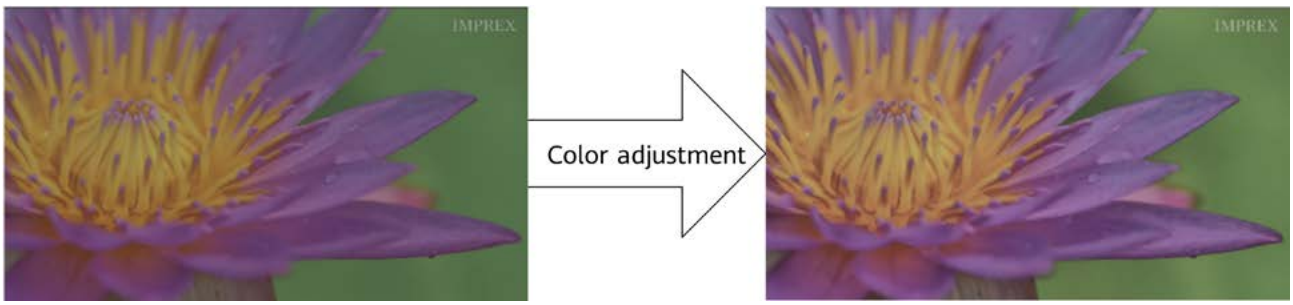


Figure 19 HDR Vivid color adjustment

3.3.3 Time Domain Stability Control

The maximum value, minimum value, average value, and variance (variation range) of luminance are the most important factors for the tone mapping curve. During display adaptation, the average luminance of the image is mapped to a position close to the medium luminance of the screen based on the variance of the luminance, the maximum luminance of the image is mapped to the highest luminance of the screen, and the minimum luminance of the image is mapped to the lowest luminance of the screen. However, HDR content has a large luminance range and includes highlights caused by diffuse reflection as well as a large number of highlights caused by specular reflection. Highlights are usually generated by light reflected on a material with a high reflectivity and are highly dependent on the angle of incidence. A slight movement or deformation of the object may change the luminance statistics of the image (as shown in Figure 20), affecting the tone mapping curve and causing a difference between two consecutive frames after tone mapping. In the research on JND [67], the distortion perceived by human eyes is correlated to the contrast sensitivity function, eye movement [68], local luminance masking effect, and contrast masking effect. If the difference between two consecutive frames is perceived by human eyes, flicker occurs [69].

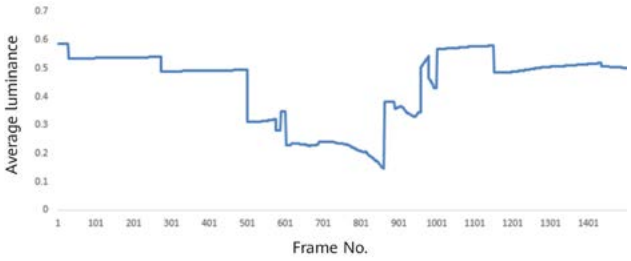


Figure 20 Time-domain variation of the average luminance in a video clip

If local tone mapping based on the frequency domain is performed, segmentation of the luminance layer and the detail layer, luminance layer mapping, and detail layer fusion during segmentation in the frequency domain may reduce the stability of the local processing result.

To improve video stability, the tone mapping curve between two consecutive frames needs to be restricted in order to eliminate time-domain oscillation. In addition, the pixel value difference between two frames caused by the change cannot exceed the minimum value that can be perceived by human eyes. Equation (31) can be used to eliminate luminance jitter in the time domain and ensure stable processing in the same scenario.

$$hdrvivid\_dy\_info\_filter = \frac{\sum_{i=0}^{hdr\_dy\_info\_fifo\_Num-1} w[i] * hdr\_dy\_info\_fifo[i]}{\sum_{i=0}^{hdr\_dy\_info\_fifo\_Num-1} w[i]} \quad (31)$$

where $hdrvivid\_dy\_info\_filter$ is the HDR Vivid information after filtering, $hdr\_dy\_info\_fifo\_Num$ is the amount of valid information about the same scenario in the HDR Vivid information, $hdr\_dy\_info\_fifo[i]$ is the previous HDR Vivid information about the same scenario and the HDR Vivid information of the current frame, and $w[i]$ is the time-domain filter kernel.

Images in the same scenario should not differ greatly from each other after processing. Otherwise, jitter or flicker may occur. The scenario detection module further identifies the scenario, such as people, plants, or animals, and provides the scenario information for the adjustment mechanism and human eye perception model to select pathways of concern. For example, the luminance of human faces must be in a normal range after processing. Otherwise, the faces will look very different from normal skin tones. The following formula can be used to further calculate whether the difference between two consecutive frames can be perceived:

$$hdrvivid\_diff = \frac{\sum_{i=0}^{HistNum} His[i] * f_{hdrvivid\_dy\_info\_filter}(L,T)}{\sum_{i=0}^{HistNum} His[i]} \quad (32)$$

where His[] indicates global or local histogram information, and $f_{hdrvivid\_dy\_info\_filter}(L, T)$ is the probability function for determining whether the tone mapping value change obtained based on the parameters after filtering can be perceived.

If the content in two consecutive frames of a video changes greatly and the content in the consecutive frames before and after the two frames is similar, a scenario switch occurs. If the content in two different scenarios cannot be separated, time domain filtering causes two consecutive frames to become darker or brighter, producing a visual effect similar to long-time flicker, as shown in Figure 21.

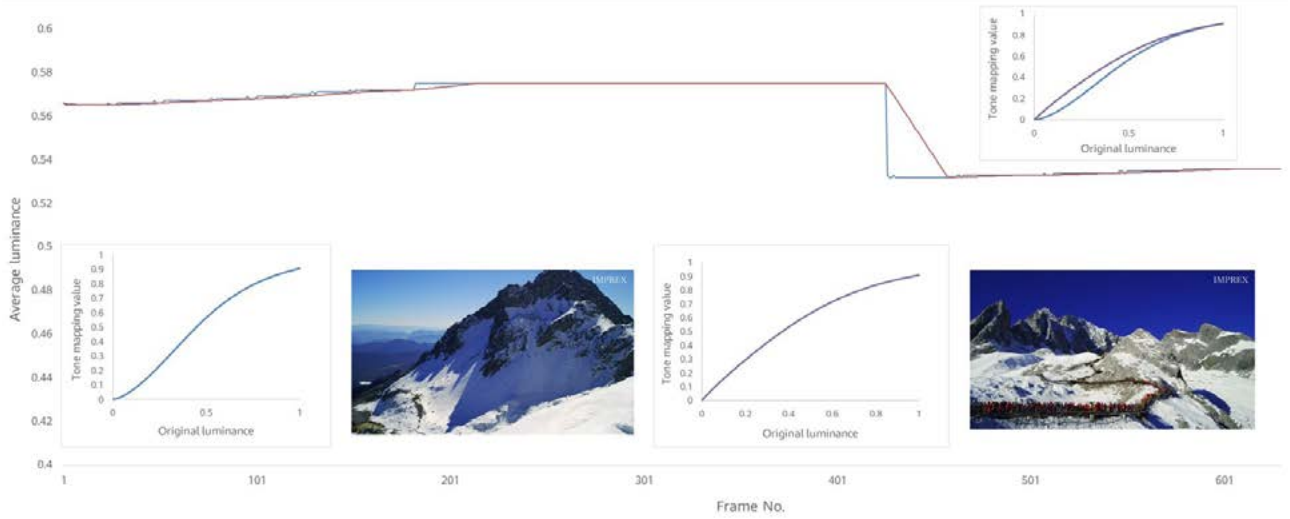


Figure 21 Comparison of time domain filtering when no scenario switch is detected

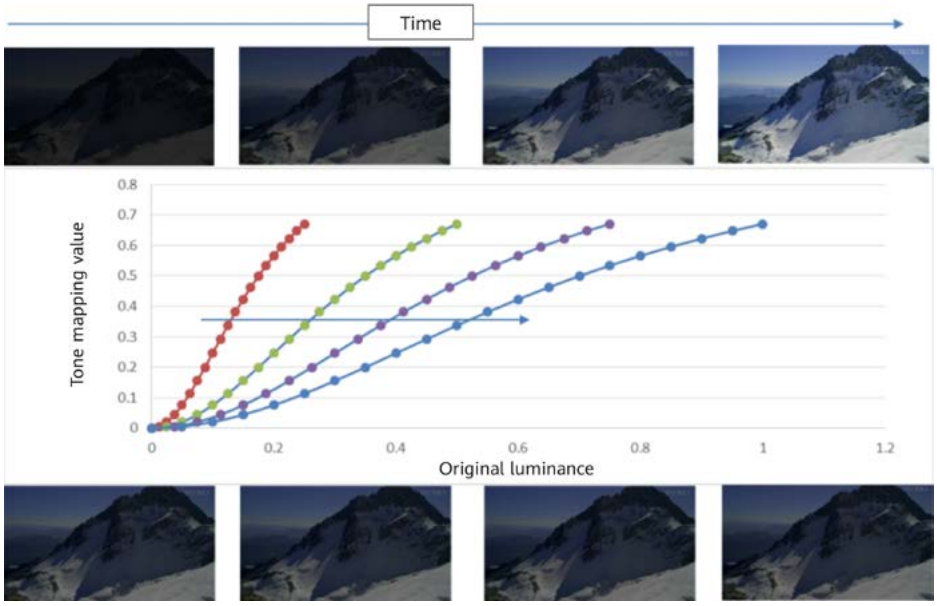


Figure 22 Tone mapping for fade-in images

Therefore, dynamic metadata extraction requires robust scenario switch detection.

In addition to time domain luminance jitter and scenario switch, the changes of bright and dark regions and gradient (fade-in and fade-out) in a video also cause time domain instability. The original luminance of different pixels in an image is different, but the values after tone mapping may be the same, resulting in a strange gradient effect, as shown in Figure 22.

This problem can be alleviated by detecting time domain changes, extending the metadata analysis window, and extending metadata generation from a single frame to multiple frames until the changes end. This method is also applicable to scenarios with changing bright and dark regions.

4 HDR Vivid Results

4.1 HDR Vivid Curve Characteristic Analysis

Two basic conditions need to be met for tone mapping processing for luminance and contrast preservation. The first one is smooth and continuous processing: $f'(x) \geq 0$ (assume that the processing function is $f(x)$). The second one is $f(x_{max}) = v_{max}$, which indicates that the maximum retained luminance of HDR content corresponds to the maximum value v_{max} of the display device. $f(x)$ has many forms but needs to be flexible to cope with different HDR scenarios. $f(x)$ also needs to be stable. To the greatest extent possible,

the image characteristics should remain unchanged after processing. Local adjustment made based on slight parameter changes between different frames should not cause large fluctuations in the image. Finally, $f(x)$ should not have many complex parameters, which may easily cause errors during the implementation by device vendors. This section analyzes the curve based on flexibility, stability, suitability, and usability.

Flexibility indicates that the curve shape may change randomly based on the HDR scenario. The curve shape is affected by many parameters.

Different parameter sets lead to different curves.
$\{f_1 \ldots f_n\} \in Ø$ indicates all valid curves.

$$Ø = \{f(x) \le v_{max};\ G'(L) \ge 0\} \tag{33}$$

The Monte Carlo method is used to randomly sample 10,000 curve parameter points and select the valid data points in space $Ø$ from all samples. The valid HDR Vivid curve can cover more space and is highly flexible.

$G_t(Y_i, p_1 \ldots p_n)$ indicates the luminance processing curve, $p_1 \ldots p_n$ are curve parameters, Y_i is the input luminance, and Y_o is the output luminance. $\Delta p_1 \ldots \Delta p_n$ indicates curve parameter changes between two frames. Specifically, $p_1 \ldots p_n$ is the curve parameter of time t, and $p_1 + \Delta p_1 \ldots p_n + \Delta p_n$ is time $t + 1$. The dynamic mapping result difference is as follows:

$$G_{t+1}(Y_i, p_1 + \Delta p_1 \ldots p_n + \Delta p_n) - G_t(Y_i, p_1 \ldots p_n) = \sum_{k=1}^{k=n} \frac{\partial G}{\partial p_k}(Y_i, p_1 \ldots p_n)\Delta p_k \tag{34}$$

The result is usually evaluated in the domain of a logarithm function. Therefore:

$$D(Y) = In(Y_i, p_1 + \Delta p_1 \ldots p_n + \Delta p_n) - In(Y_i, p_1 \ldots p_n)$$

$$= \frac{1}{G_t(Y_i, p_1 \ldots p_n)} \sum_{k=1}^{k=n} \frac{\partial G}{\partial p_k}(Y_i, p_1 \ldots p_n)\Delta p_k \tag{35}$$

The stability is calculated as follows:

$$S(Y) - 1 - \int_{Ymin}^{Ymax} D(Y)dY \tag{36}$$

If the input and output are normalized to [0, 1], the value of $S(Y)$ is also in [0, 1]. The maximum value 1 indicates the highest stability, and the minimum value 0 indicates the lowest stability. Figure 24 shows the curve changes with each 10-nit increase of luminance and the curve stability

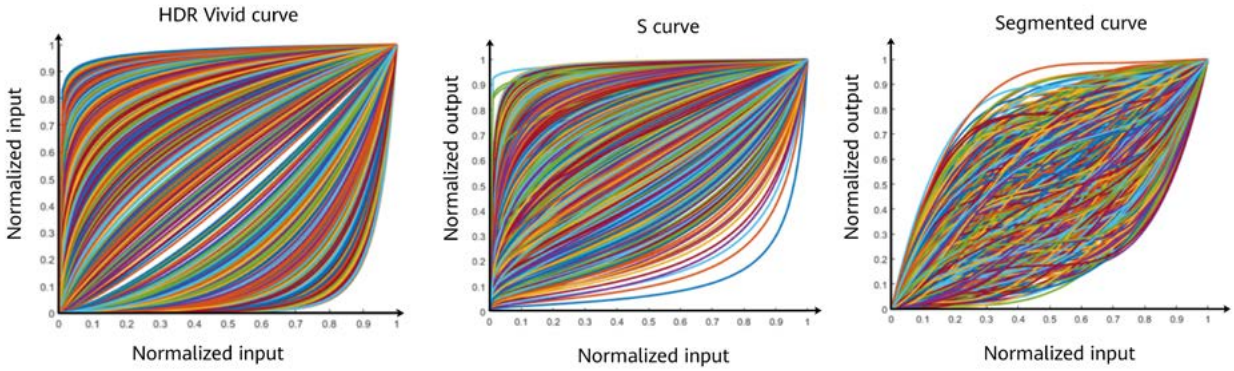


Figure 23 Effective curve space based on Monte Carlo method

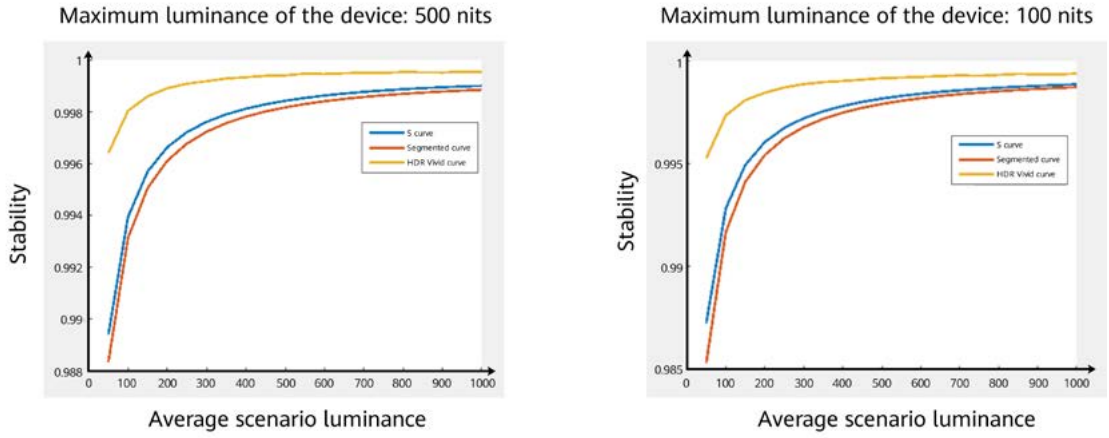


Figure 24 Relationship between stability and average scenario luminance

at the target luminance of 500 nits and 100 nits. The HDR Vivid curve is more stable.

The distribution features of the video after luminance processing should be the same as those of the original video in order to ensure a natural visual effect. Therefore, the proposed curve, S curve, and segmented curve are used to process the random sample points on the spline curves with an input that follows Gaussian distribution, and obtain the output values.

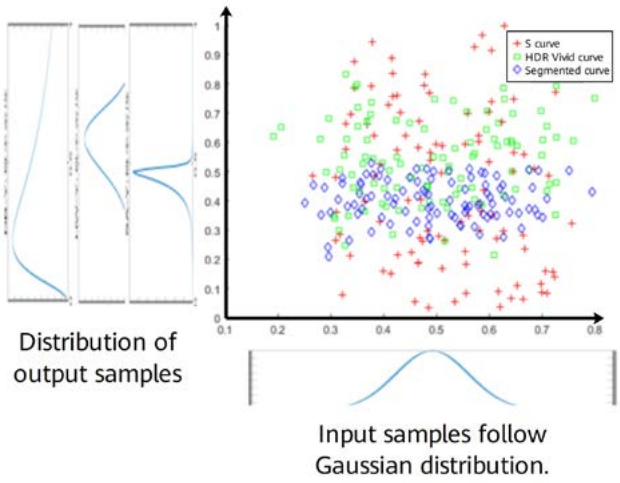


Figure 25 Scatter diagram of different curves

Both the input and an output are normalized to [0, 1.0], as shown in Figure 25. We can see that the shape of the output spline curve generated based on the proposed curve is most similar to the curve shape of the input sample points. The S curve modifies the input sample points, which changes the overall display effect. The segmented curve limits the distribution of input sample points within a smaller range, reducing the probability of values in some regions. Figure 26 shows the histograms of different curves. The horizontal coordinates are linear values. The vertical coordinates are the distributions corresponding to the linear values. The upper four histograms follow single-peak Gaussian distribution, and the lower four follows dual-peak Gaussian distribution. The histograms of the S curve and the proposed curve are most similar to the original distribution. The S curve extends the distribution to a larger range, while the segmented curve reduces the distribution to a smaller range. The proposed curve better maintains the features of the original video, accurately reproducing the artistic intentions of creators.

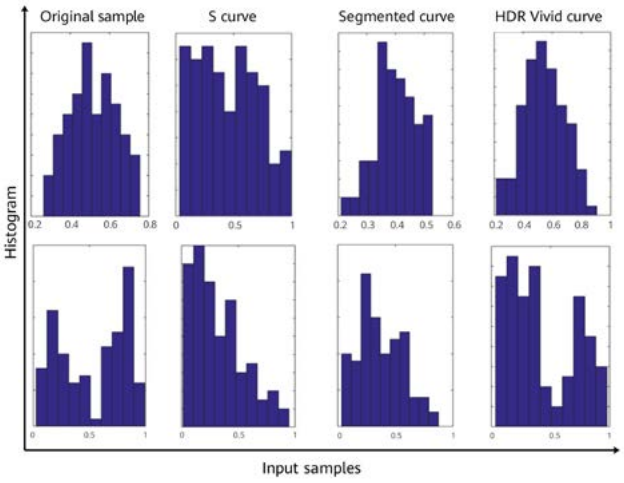


Figure 26 Histograms of different curves

4.2 HDR Vivid Curve Effect Analysis

An HDR video is used to test the HDR Vivid effect on different display devices. The peak luminance of the HDR video ranges from 500 nits to 10,000 nits, the minimum luminance is less than 0.001 nit, the color gamut is within the range defined by BT.2020. The test display devices include both SDR and HDR devices. The peak display luminance of the SDR devices is 100 nits, the output format is gamma 2.2, and the color gamut is the one defined by BT.709. The peak luminance of one HDR device is 400 nits and that of the other device is 1000 nits, the output format is PQ, and the color gamut is DCI-P3. The process of generating HDR Vivid metadata in section 3.3.1 is used to analyze the impact of the final mapping effect based on the selected luminance feature points. First, the overall contrast is analyzed. The overall mapping effect depends on the two luminance feature points in the middle, which form the basic curve. m_a and m_p are key parameters. For horizontal coordinates of two luminance feature points, the interval between the vertical coordinates affects the contrast of the entire image. In Figure 27 1a, m_p is large and m_a is small, the basic curve is flat, and the overall contrast is low. In Figure 27 2a, m_p decreases, and m_a increases, improving the overall contrast. The shape of the mapping curve in Figure 27 3a is suitable, maintaining the original contrast to the maximum. Figure 27 1b, 2b, and 3b are the display effects of these mapping curves.

The luminance of dark regions depends on the first luminance feature point. According to section 3.3.1, the horizontal coordinate of luminance feature point 1 depends

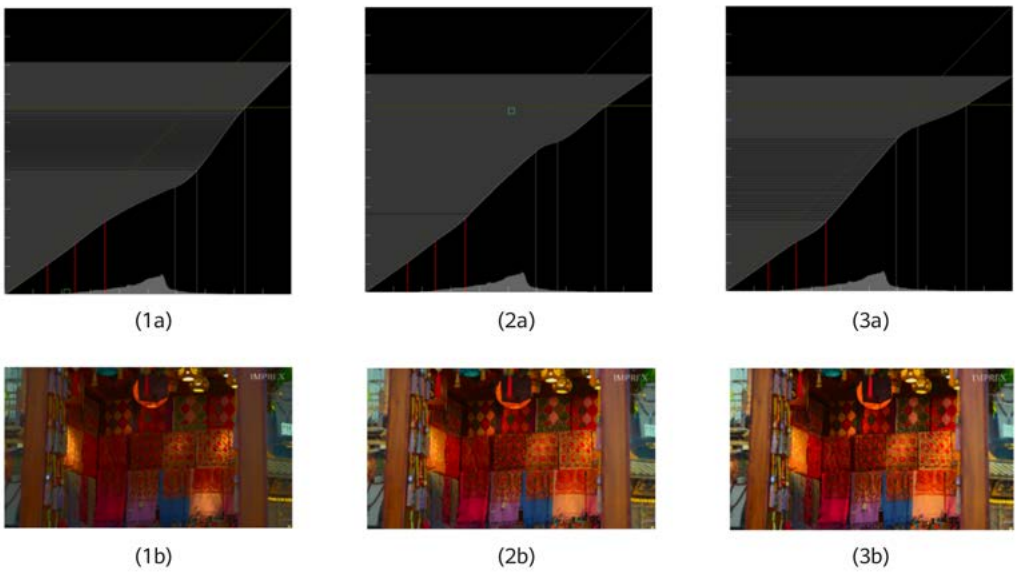


Figure 27 Impact on the overall contrast on a 400-nit HDR device

on the lower limit of luminance perceived by human cones in previous research. Determining the vertical coordinate, which is the slope of the primary spline curve, is key. In Figure 28 1a, the primary spline curve covers a large number of pixels, and the shape of the mapping curve is more suitable. The slope in Figure 28 2a is small (the vertical coordinate of luminance feature point 1 is small), resulting in overcompression and loss of many details. Figure 28 1b and 2b are the display effects and local details of the two mapping curves.

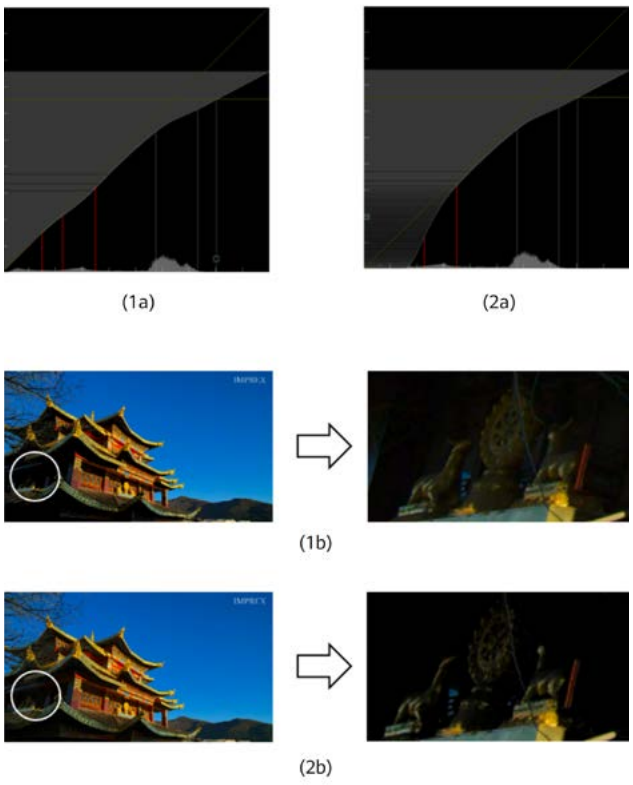


Figure 28 Impact of the slope of the primary spline curve on the luminance of dark regions on a 400-nit HDR device

Similar to details in bright regions, the display effect of details in dark regions depends on the vertical coordinate of the middle point on the two cubic spline curves. Changing the offset value and direction affects the contrast and details of dark and bright regions. Take details in bright regions as an example. Figure 29 1a shows the pixel quantity relationship of the luminance histograms in the two intervals between the last three luminance feature points. A negative (downward) offset is added to the initial value of the vertical coordinate of the middle luminance feature point, improving the contrast and details of bright regions in comparison with Figure 29 2a, where no adjustment is performed. An incorrect offset direction has a negative impact on the display effect. For example, a positive (upward) offset with the same amount is added in Figure 29 3a, producing a worse effect in comparison with Figure 29 2a. Figure 29 1b, 2b, and 3b are the display effects and local details of the three mapping curves.

Highlight preservation is also key to tone mapping. HDR Vivid carefully selects the horizontal coordinate of the last luminance feature point based on the maximum display capability of a device to preserve highlight information. The vertical coordinate is set to the maximum luminance of the device. As shown in Figure 30 1a, the horizontal coordinate of luminance feature point 6 is the maximum luminance of the frame, retaining all highlight information after mapping. The horizontal coordinate of luminance feature point 6

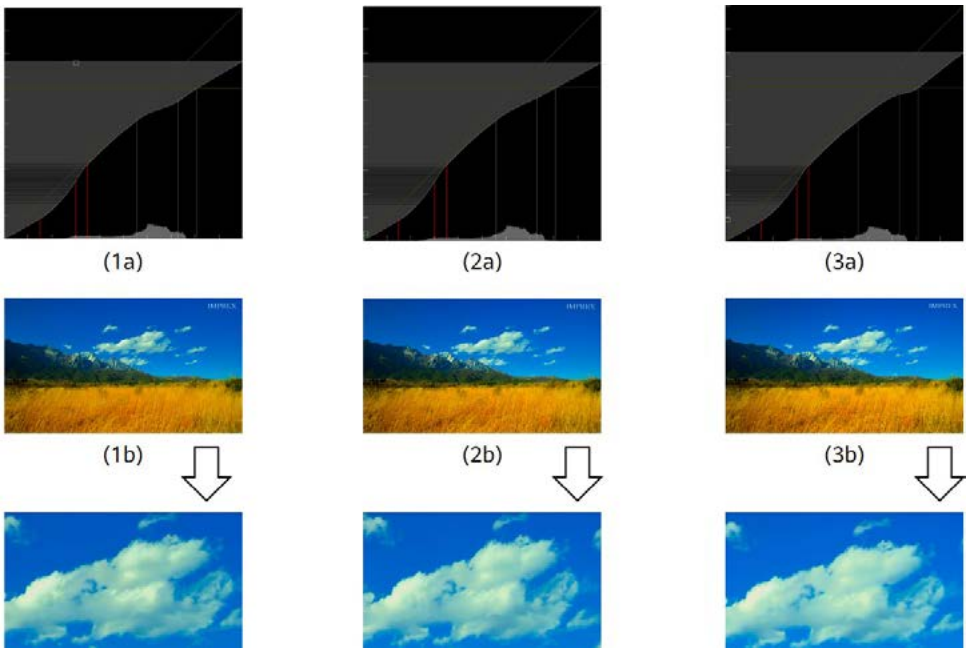


Figure 29 Impact of the vertical coordinate of the middle luminance feature point on the cubic spline curves on details in bright regions on a 400-nit HDR device

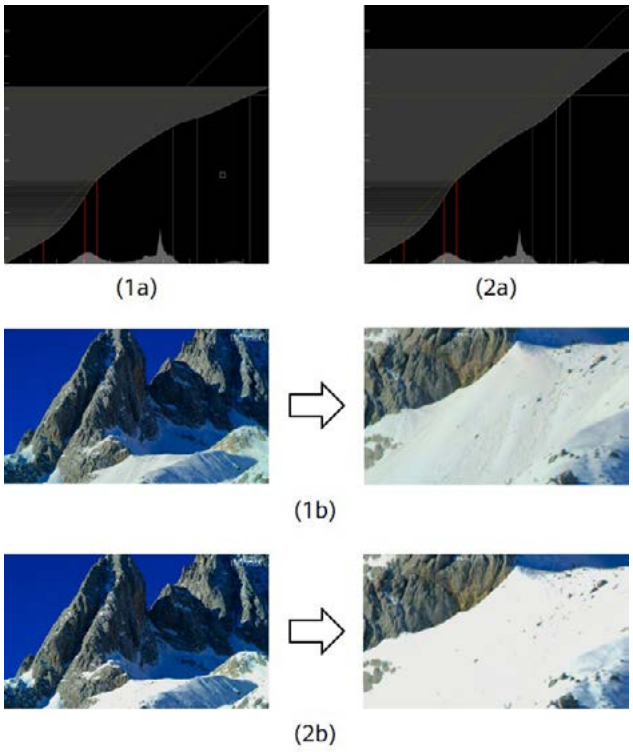


Figure 30 Impact of the horizontal coordinate of the last luminance feature point on highlights on a 400-nit HDR device

Figure 31 HDR Vivid color preservation comparison

in Figure 30 2a is small, causing the luminance of a large number of pixels to exceed the capability of the display device after mapping. This phenomenon is overexposure, which causes loss of highlight information. Figure 30 1b and 2b are the display effects and local details of the two mapping curves.

HDR Vivid uses color compensation technology to maintain the original colors. As shown in Figure 31, other solutions cause a large color deviation and excessive saturation, while HDR Vivid better maintains the original colors.

4.3 HDR Vivid E2E Effect Comparison

Five common tone mapping methods [70–74] are compared, and the optimal display effects of these methods are selected for each scenario and compared with HDR Vivid, as shown in Figure 32 to 34. In Figure 32, the original image has many details in dark regions and a high average luminance. HDR Vivid better reproduces details in dark regions while maintaining the luminance of the main body. This is because HDR Vivid designs a special spline function to reproduce details in dark regions during luminance processing. In Figure 33 and 34, the original image has many highlight details and a high average luminance. HDR Vivid better restores the highlight details and maintains the contrast of the main body. This is because HDR Vivid uses an inverted S curve to process highlights and an S curve to process the main body.

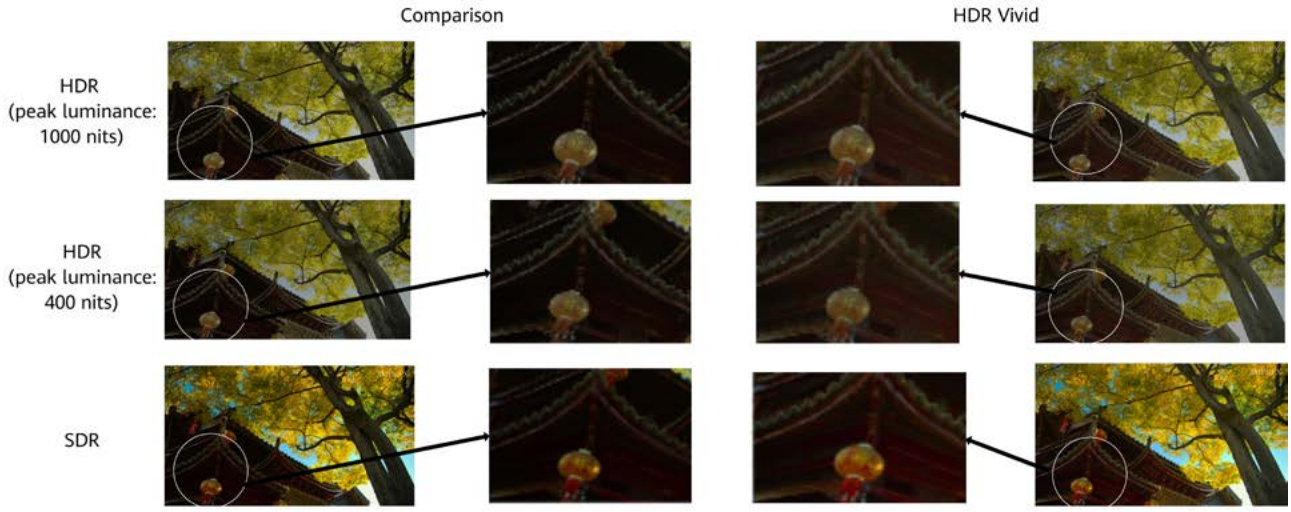


Figure 32 Loss of details in dark regions for HDR Vivid and other methods

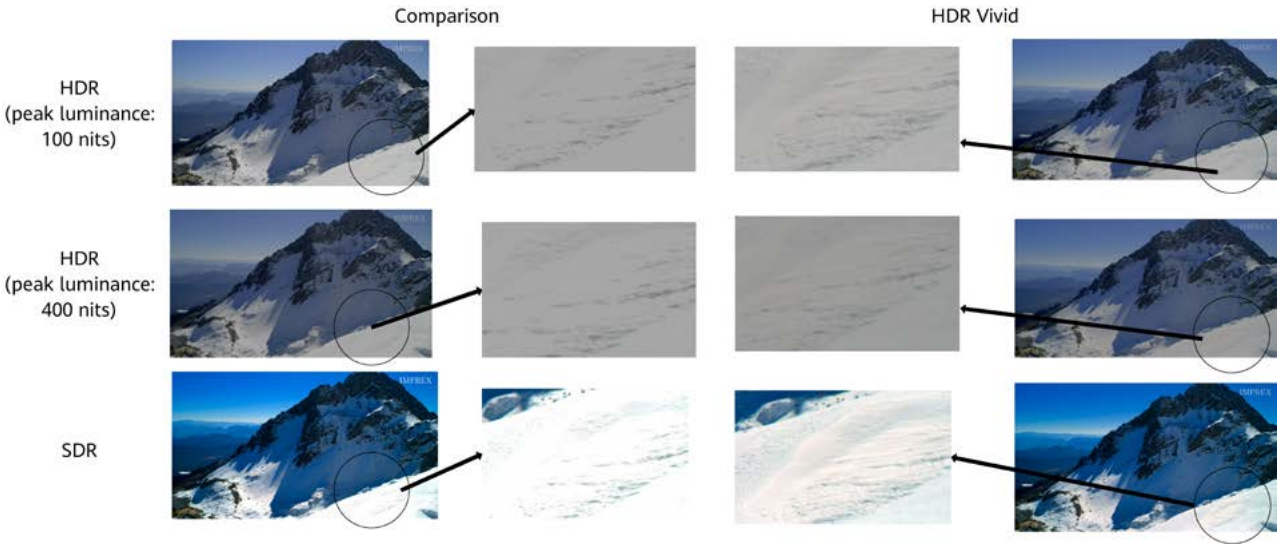


Figure 33 Loss of details in bright regions for HDR Vivid and other methods

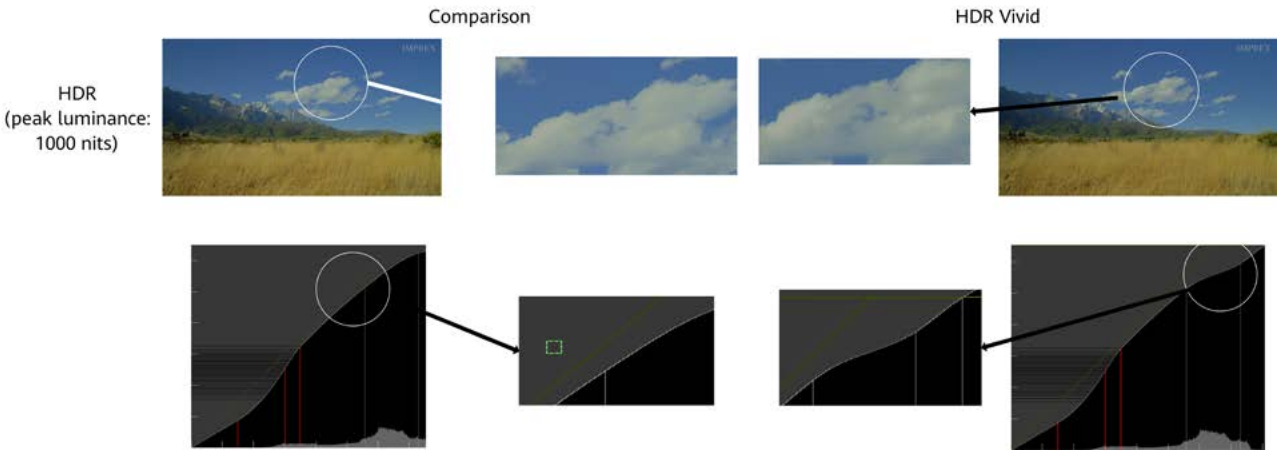


Figure 34 Highlight details of HDR Vivid and other methods

5 Conclusion

This paper describes the HDR Vivid standard, principles, and technology. HDR Vivid is an E2E solution that streamlines production, transmission, display, and perception to reproduce the artistic intentions of creators for HDR Vivid device users. This solution allows creators to use more flexible techniques to create content, without worrying about the display effect on different devices. Specifically, the HDR Vivid technology leverages the characteristics of the visual system to ensure consistent display effects on different devices. HDR Vivid presents the richest colors, finest details, stronger contrast, vivid gradient, and more dimensions on various display devices, such as TVs, computers, mobile phones, and tablets, delivering a superior viewing experience.

References

[1] *Photography – digital still cameras – guidelines for reporting pixel-related specifications*. ANSI/I3A IT10.7000–2004.

[2] *Recommendation ITU-R BT.709-6: Parameter values for the HDTV standards for production and international programme exchange*.

[3] *Recommendation ITU-R BT.2020-2: Parameter values for ultra-high definition television systems for production and international programme exchange*.

[4] G.J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand (May 25, 2012), "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, Retrieved May 18, 2013.

[5] Sean Whaley (21 November 2018), "What is frame rate and why is it important to PC gaming?," Retrieved 5 August 2021.

[6] Thomas De Quincey (1854), *De Quincey's works*. James R. Osgood. p. 36. gamut-of-hues 0-1856.

[7] "Dynamic range", *Electropedia*, IEC, archived from the original on 2015-04-26.

[8] *Recommendation ITU-R BT.2100-2: Image parameter values for high dynamic range television for use in production and international programme exchange*.

[9] https://alliance.experienceuhd.com/uhd-premium-features.

[10] L. Meylan, "Tone mapping for high dynamic range images," *EPFL*, Tech. Rep., 2006.

[11] E. Reinhard and K. Devlin, "Dynamic range reduction inspired by photoreceptor physiology," *IEEE Trans*. Vis. Comput. Graphics, vol. 11, no. 1, pp. 13–24, Jan. 2005.

[12] M. H. Kim and J. Kautz, "Consistent tone reproduction," in Proc. 10th IASTED Int. Conf. Comput. Graphics Imaging, D. Thalmann, Ed. Anaheim, CA, USA: ACTA, 2008, pp. 152–159.

[13] V. A. Patel, P. Shah, and S. Raman, "A generative adversarial network for tone mapping HDR images," in Proc. 6th Nat. Conf. Comput. Vis., Pattern Recog., Image Process. Graph. (NCVPRIPG), Apr. 2018, pp. 220–231.

[14] Z. Zhang, C. Han, S. He *et al*., "Deep binocular tone mapping," Vis. Comput., vol. 35, nos. 6–8, pp. 997–1011, Jun. 2019.

[15] ST 2086:2014 - *SMPTE Standard - Mastering Display Color Volume Metadata Supporting High Luminance and Wide Color Gamut Images*.

[16] Wu Ying, "Yishu guanli yu shichang" [Art management and market], Communication University of China Publishing House, 2017: p. 106.

[17] Guo Maolai, "Shijue yishu gailun" [An overview of visual arts], People's Fine Arts Publishing House, 2000, p. 12

[18] Patrick Frank, *Art forms: an introduction to the visual arts (8th Edition), Pearson Education, 2005*

[19] Patrick Frank, *History of visual art*

[20] Arnason, H.H.H and Peter.Kalb, *History of Modern Art*

[21] Porta, *Natural Matic*, 1558

[22] https://www.ilfordphoto.com/amfile/file/download/file/1748/product/735/?___store=ilford_brochure&___from_store=ilford_brochure.

[23] Mees, C.E.K. *The Theory of the Photographic Process. Macmillan*, 1942: 542-551.

[24] https://www.youtube.com/watch?v=idepJM8iHpY&t=54s.

[25] https://link.springer.com/article/10.3758/s13414-012-0281-4.

[26] https://en.wikipedia.org/wiki/Weber%E2%80%93Fechner_law.

[27] HUNT, "R. W. G., light energy and brightness sensation," https://libgen.is/scimag/10.1038%2F1791026a0.

[28] Stevens, S. S. (1957). "On the psychophysical law," *Psychological Review*, 64(3), 153–181.

[29] https://coggle.it/diagram/V-hiK5vlWIhPCd_F/t/weber's-and-devries-rose-law.

[30] P.G. J. Barten, "Physical model for the Contrast Sensitivity of the human eye," Proc. SPIE 1666, 57-72 (1992).

[31] ITU-R BT.2246-5: *The present state of ultra-high definition television*.

[32] PS 3.14-2001. "Part 14: grayscale standard display function," *Digital Imaging and Communications in Medicine (DICOM)*.

[33] Aydın, T. O., Mantiuk, R., and Seidel, H.-P. (2008), "Extending quality metrics to full luminance range images," *Human Vision and Electronic Imaging XIII*.

[34] *Society of Motion Pictures and Television Engineers (SMPTE)*. SMPTE ST 2084:2014 – High Dynamic Range Electro-Optical Transfer Function of Mastering Reference Displays. Technical report, Society of Motion Pictures and Television Engineers (SMPTE), White Plains, NY, August 2014.

[35] "High dynamic range," *European Broadcasting Union*. Retrieved 2015-11-01.

[36] Artal P, "Image formation in the living human eye," *Annual Review of Vision Science*, 2015, 1: 1–17.

[37] Lee BB, Martin PR, and Grünert U, "Retinal connectivity and primate vision," *Progress in Retinal and Eye Research*, 2010, 29(6): 622–39.

[38] Mante V, Frazor RA, Bonin V, Geisler WS, and Carandini M. "Independence of luminance and contrast in natural scenes and in the early visual system," *Nature Neuroscience*, 2005, 8(12): 1690.

[39] Valeton J and Norren DV, "Light adaptation of primate cones: an analysis based on extracellular data," *Vision Research*, 1983, 23(12): 1539–47.

[40] Zicong Mai, "Optimizing a tone curve for backward-compatible high dynamic range image and video compression," *IEEE Transactions on Image Processing*, Institute of Electrical and Electronics Engineers, 2017, ff10.1109/TIP.2017.

[41] DALY, S. 1993, "The visible differences predictor: an algorithm for the assessment of image fidelity." in

Digital Image and Human Vision, Cambridge, MA: MIT Press, A. Watson, Ed., 179–206.

[42] KUANG, J., JOHNSON, G. M., AND FAIRCHILD, and M. D. 2007, iCAM06: "A refined image appearance model for HDR image rendering," *Journal of Visual Communication and Image Representation*, 18, 5, 406–414.

[43] HUNT, R. 2004, "The reproduction of colour in photography," *Printing and Television: 6th Edition*. John Wiley & Sons.

[44] Rafal Mantiuk, Kil Joong Kim, Allan G.Rempel, and Wolfgang Heidrich, HDR-VDP-2: "A calibrated visual metric for visibility and quality predictions in all luminance conditions," in *ACM Transactions on Graphics*, article no. 40, 2011.

[45] E. Reinhard and K. Devlin, "Dynamic range reduction inspired by photoreceptor physiology," *IEEE Trans*, Vis. Comput. Graphics, vol. 11, no. 1, pp. 13–24, Jan. 2005.

[46] M. H. Kim and J. Kautz, "Consistent tone reproduction," in Proc. 10th IASTED Int. Conf. Comput. Graphics Imaging, D. Thalmann, Ed. Anaheim, CA, USA: ACTA, 2008, pp. 152–159.

[47] L. Lenzen, M and Christmann, "Subjective viewer preference model for automatic HDR down conversion," *Electron. Imaging*, vol. 12, pp. 191–197, 2017.

[48] Mante V, Frazor RA, Bonin V, Geisler WS, and Carandini M, "Independence of luminance and contrast in natural scenes and in the early visual system," *Nature Neuroscience*, 2005, 8(12): 1690.

[49] Billock VA and Tsou BH, "To honor Fechner and obey Stevens: relationships between psychophysical and neural nonlinearities," *Psychological Bulletin*, 2011, 137(1):1.

[50] Radonjic A, Allred SR, Gilchrist AL, and Brainard DH, "The dynamic range of human lightness perception," *Current Biology*, 2011, 21(22): 1931–6.

[51] Campbell FW and Robson J, "Application of Fourier analysis to the visibility of gratings," *The Journal of Physiology*, 1968, 197(3): 551–66.

[52] Barten PG, "Contrast sensitivity of the human eye and its effects on image quality," vol. 21. Bellingham, WA: Spie Optical Engineering Press; 1999.

[53] Whittle P, "Brightness, discriminability and the "crispening effect"," *Vision Research*, 1992, 32(8): 1493–507.

[54] Nundy S and Purves D, "A probabilistic explanation of brightness scaling," *Proceedings of the National Academy of Sciences*, 2002, 99(22): 14482–7.

[55] D. C. Hood and M. A. Finkelstein, *Handbook of Perception & Human Performance*, Wiley, 1986.

[56] Jack Tumblin and Greg Turk, "LCIS: a boundary hierarchy for detail-preserving contrast reduction," *Computer Graphics Proceedings*, SIGGRAPH 99, pp. 83-90.

[57] Li. Y., Sharan and L.m Adelson, E., "Compressing and companding high dynamic range images with subband architectures," *ACM trans*. Graph.23(3). 2005.

[58] Lubbe, E., *Colours in the Mind - Colour Systems in Reality*, 2008.

[59] Hunt, "R. W. G., a model of colour vision for predicting colour appearance," *Color Res*. Appl. 7, 95-112. 1982.

[60] Pearson, K. (1895), "Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 186: 343–414.

[61] T. L. Kong and N. A. Mat Isa, "Bi-histogram modification method for nonuniform illumination and low-contrast images," *Multimedia Tools Appl.*, vol. 77, no. 7, pp. 8955–8978, 2018.

[62] S. Fong Tan and N. Ashidi Mat Isa, "Exposure based multi-histogram equalization contrast enhancement for non-uniform illumination images," *IEEE Access*, vol. 7, pp. 70842–70861, 2019.

[63] N. H. Saad, N. A. M. Isa, and A. A. M. Salih, "Local neighbourhood image properties for exposure region determination method in nonuniform illumination images," *IEEE Access*, vol. 8, pp. 79977–79997, 2020.

[64] Acharya and Ray, *Image Processing: Principles and*

Applications, Wiley-Interscience 2005 ISBN 0-471-71998-6.

[65] *Recommendation ITU-R BT2408-3: Guidance for operational practices in HDR television production*.

[66] Rafał Mantiuk, Scott Daly, Louis Kerofsky, "Display adaptive tone mapping," *ACM Transactions on Graphics*, Vol. 27, No. 3, Article 68, Publication date: August 2008.

[67] Jia Y T, Lin W S, and Kassim A A, "Estimating just-noticeable distortion for video," *IEEE Transactions on Circuits and Systems for Video Technology*, 2006, 16(7): 820-829.

[68] Recasens, A., Khosla, A., Vondrick, C., and Torralba, A., *Where are they looking?*, NIPS 2015.

[69] B. Guthier, S. Kopf, M. Eble, and W. Effelsberg, "Flicker reduction in tone mapped high dynamic range video," SPIE, 2011.

[70] R Mantiuk, K Myszkowski, and HP Seidel, "A perceptual framework for contrast processing of high dynamic range images," *ACM Transactions on Applied Perception (TAP)*, 3 (3), 286-308.

[71] S. Ferradans, M. Bertalmio, E. Provenzi, and V. Caselles, "An analysis of visual adaptation and contrast perception for tone mapping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10): 2002–2012, Oct 2011.

[72] Durand F and Dorsey J, "Fast bilateral filtering for the display of high-dynamic-range images" [J]. *ACM Transactions on Graphics*, 2002, 21(3): 257-266.

[73] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans*. Graph, 21(3): 257–266, 2002.

[74] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," *ACM Trans*. Graphics, 21(3): 249–256, July 2002.

OpenEuler

Open Source Operating System for Digital Infrastructure

Supporting diversified computing and meeting all-scenario requirements of servers, cloud, edge, and embedded devices

100% Support for Mainstream Computing Architectures

ARM, x86, RISC-V, SW-64, LoongArch, NPU, GPU, DPU, 100+ devices, and 300+ boards

Support for Diversified Computing

Information Technology | Communication Technology | Operational Technology

CRM | ERP | BSS/OSS | NFV | DCS | SCADA | PLC | .....

100% Support for Mainstream Application Scenarios

Cloud native | Big data | CDN | MEC | Industrial control | ...

All-Scenario Applications

openEuler DevKit

System Build and Deployment

- Multi-device build tool
- Compatibility/Porting/Optimization tool

openEuler SDK

All-Scenario Application Development

- Multi-architecture confidential computing SDK
- Application compatibility assessment SDK

Complete Development Tool Chain

openEuler official WeChat account

openEuler official website

HUAWEI TECHNOLOGIES CO., LTD.
Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129, PRC
Tel: +86-755-28780808