

Toward a Self-Evolving Optimization Agent for Industrial Scheduling

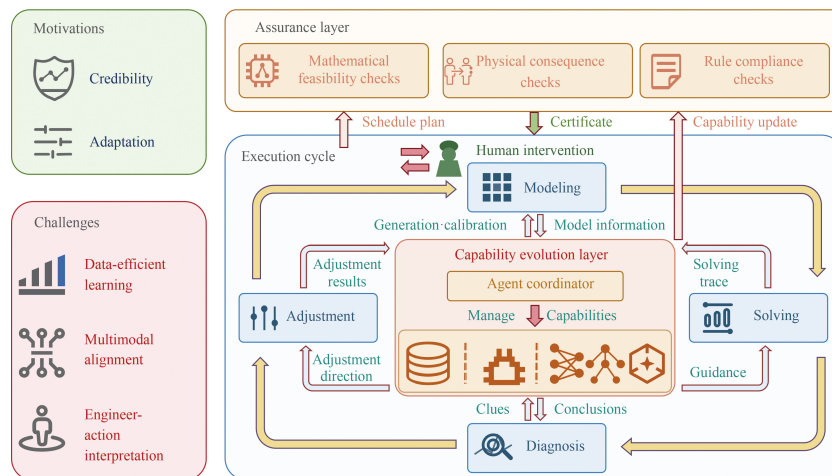
Tianao Kang^{1,2}, Luolin Xiong^{1,2}, Wei Du^{1,2}, Yang Tang (✉)^{1,2}, Feng Qian (✉)^{1,2}

¹ Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China

² State Key Laboratory of Industrial Control Technology, East China University of Science and Technology, Shanghai 200237, China

© The Author(s) 2026. This article is published by Higher Education Press.

GRAPHICAL ABSTRACT



HIGHLIGHTS

- A three-component architecture connects task execution, capability evolution, and assurance for industrial scheduling.
- Scheduling records support controlled updates to algorithms and their conditions of use.
- Key challenges include limited data, multimodal alignment, and interpreting engineer actions.

ABSTRACT AI methods for industrial scheduling have shown considerable promise, yet two obstacles hinder their deployment in real production: credibility and adaptability. Strong laboratory performance does not guarantee that a plan is fully safe and reasonable in production, so human review remains necessary, and the deployment and improvement of algorithms still depend heavily on engineer experience. This paper highlights a self-evolving optimization agent for these problems. The agent automates modeling, solving, diagnosis, and adjustment for each task through an Execution Cycle, and uses information from repeated scheduling work through a Capability Evolution Layer to improve the performance and conditions of use of the algorithms supporting the workflow, thereby achieving automatic algorithm adaptation. For credibility, the system employs an Assurance Layer to check each execution plan and proposed algorithm change through mathematical feasibility checks, physical consequence checks in represented simulation scenarios, and compliance checks for explicitly formalized operating and safety requirements. Furthermore, we identify three research challenges in realizing such a system: learning from limited data, aligning multimodal information with the same production system, and identifying the effects and implications of engineer actions, which provide valuable guidance for building the system.

KEYWORDS Self-evolving optimization; Industrial scheduling; Continual learning; Mixed-integer programming; Solver-in-the-loop; Human-AI collaboration

Received 31 Jul 2026; accepted 8 Oct 2026

E-mails: yangtang@ecust.edu.cn (Tang Y.),
fqian@ecust.edu.cn (Qian F.).

1 Introduction

AI methods have shown promising results on many scheduling problems, yet there remains a distance before these algorithms can operate automatically and credibly in real plants. Currently, a typical industrial scheduling task proceeds through modeling, solving, diagnosis, and adjustment. Automation is most mature in solving, while engineers remain indispensable for encoding constraints and calibrating parameters during modeling, inspecting solver records and investigating infeasibility during diagnosis, and applying knowledge of equipment, operating limits, and production requirements during adjustment. Notably, each task also generates valuable information that can inform later algorithmic improvement; however, in real production settings, because of the burden of data curation, such information is rarely organized for that purpose.

Recent advances in AI for optimization have shown potential in several parts of the solving process. Learned methods have supported solver decisions such as variable selection, initialization, local search, relaxation, and cutting-plane generation across instances [1–3]. Prediction-correction frameworks bring AI-generated solutions into closer agreement with real-world constraints [4]. Lifelong learning and self-evolution methods study how optimization strategies can improve over sequences of tasks [5,6]. Instance-wise adaptation addresses the use of task-specific information during solution generation [7]. These studies demonstrate that AI methods can not only effectively improve scheduling performance, but also hold the potential to continuously improve the algorithms' own capabilities. However, many of these seemingly attractive studies remain at the laboratory and benchmark level, whereas industrial scheduling involves more variable, more safety-critical, and more complex scenarios, which makes the practical deployment of AI algorithms difficult.

Two obstacles explain the difficulty of deploying AI scheduling algorithms in industrial scheduling. The first is a credibility barrier. Research on AI for optimization often reports solver speed, objective values, or benchmark solution quality. However, an industrial plan must also satisfy the mathematical constraints represented in the model and produce acceptable consequences under the relevant operating conditions. Engineering review therefore remains central to AI-supported scheduling decisions. The second is an adaptability bottleneck. An algorithm must be configured for the task in which it is used, and its performance can depend on choices made at deployment time. Several challenges are difficult to avoid in industrial settings: production requirements and objectives contain aspects that are difficult to express

numerically. An algorithm may therefore optimize a workable proxy while producing a plan that conflicts with an engineer's judgment about the production situation. In addition, a data-driven method trained on benchmark instances or another production context may degrade when the current conditions differ from its training distribution. Adaptability is thus critical for an algorithm to deliver its expected performance, and at present it depends heavily on human experience.

Research in continual learning and automated machine learning (AutoML) offers methods for updating, selecting, and configuring algorithms as tasks change [8,9]. Industrial scheduling, however, also requires revising mathematical optimization models, coordinating modeling, solving, diagnosis, and adjustment, and governing when updated capabilities can be reused. Improved algorithmic performance alone does not establish that automatically generated changes will produce acceptable plans under the represented constraints and operating conditions.

These two obstacles call for a self-evolving optimization agent that can execute complete scheduling tasks automatically and credibly, while fully exploiting the self-improvement capacity of algorithms by improving algorithmic performance from information generated during actual scheduling work. Based on the high reliability requirements of industrial scheduling and the specific content of scheduling tasks, we discuss how to structure such a self-evolving system and which challenges must be addressed. This paper describes the architecture in Section 2, identifies three challenges that must be addressed for data-driven self-improvement in Section 3, and discusses the research value and requirements of this direction in Section 4.

2 The Architecture of a Self-Evolving Optimization Agent

As illustrated in Fig. 1, the architecture has three components with distinct responsibilities. The Execution Cycle automates one scheduling task through modeling, solving, diagnosis, and adjustment. The Capability Evolution Layer improves the performance and conditions of use of the algorithms connected to this cycle by learning from repeated scheduling work. The Assurance Layer checks each execution plan and proposed algorithm change against the mathematical constraints, simulation results, and formally represented operating rules. Together, these components connect task execution, data-driven algorithm improvement, plan verification, and controlled capability updates.

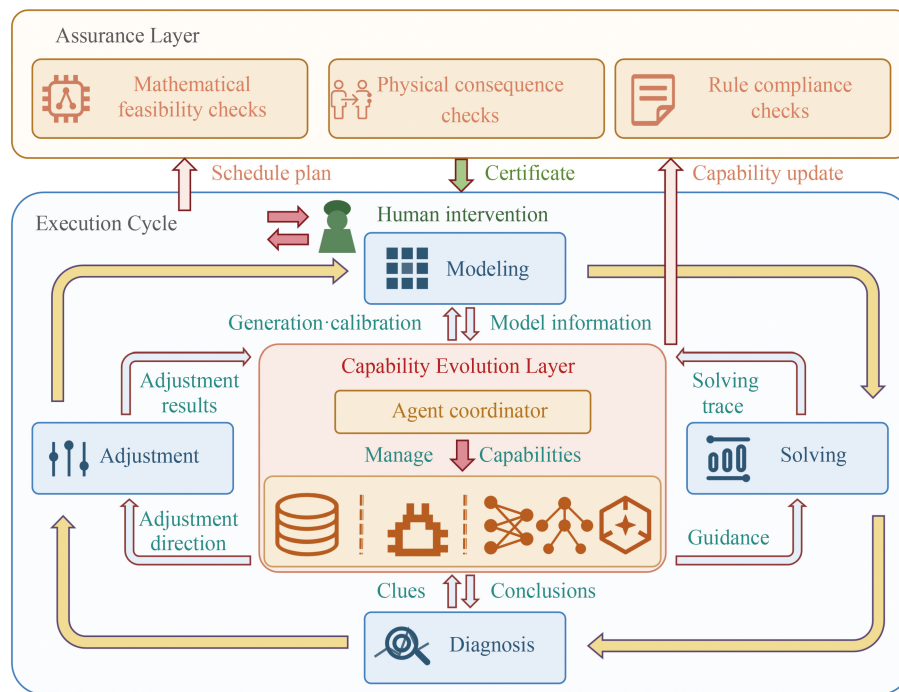


Fig. 1 Architecture of a self-evolving optimization agent for industrial scheduling. The Execution Cycle completes modeling, solving, diagnosis, and adjustment for each scheduling task. The Capability Evolution Layer manages the algorithms used in these stages and improves their selection, configuration, and use from information generated during scheduling work. The Assurance Layer evaluates plans and algorithmic modifications through mathematical feasibility checks, physical consequence checks, and rule-compliance checks.

2.1 The Execution Cycle

The Execution Cycle completes one scheduling task through modeling, solving, diagnosis, and adjustment. It provides the setting in which algorithms are applied to production-related conditions and in which the resulting plans, computational records, interventions, and operating feedback become evidence for later improvement.

At the modeling stage, the agent can support model construction and revision by using process mechanisms, engineering documents, and historical modeling decisions. At the solving stage, it can apply algorithmic methods that improve computational performance and solution quality, including learned search guidance and solver hyper-parameter configuration. At the diagnosis stage, it can relate solver information to diagnostic records to identify infeasibility sources and recurring performance bottlenecks. For example, a language model may help interpret diagnostic records to provide debugging suggestions supported by a knowledge base. The diagnostic results inform the adjustment stage; depending on their form and confidence, the resulting changes are accepted automatically after Assurance Layer verification or after engineer review. At the adjustment stage, the agent can automatically formulate and implement approved changes to the model or algorithm,

or prepare other higher-level change suggestions for engineer review. Each execution plan is passed to the Assurance Layer for checking before automatic application.

2.2 The Capability Evolution Layer

The Capability Evolution Layer is the core of self-evolution; it manages the algorithmic capabilities connected to the scheduling workflow. An algorithmic capability consists of the algorithm itself, its configuration interface, and its knowledge base. The Agent Coordinator uses the task conditions and the available capability records to determine how a capability should be used for the current task.

The purpose of this layer is performance improvement grounded in real scheduling data. A task produces evidence at every stage, including computational performance, plan quality, constraint satisfaction, engineer interventions, and other operating feedback. The Capability Evolution Layer uses this feedback to judge whether a capability performed adequately, where its performance degraded, and what kind of change may improve its future use. When an algorithm is judged to require improvement, the agent updates it through the update pathway reserved by the capability in the form of its configuration interface. Depending on the type of the

algorithm, an update may involve updating learned parameters, refining decision rules, changing a model component, changing a solver configuration, updating the knowledge about the conditions under which the capability is applied, or other modifications. Various techniques can be used to provide update interfaces for algorithms, such as Bayesian optimization, online learning, and other algorithms.

In industrial scheduling, where credibility is a central concern, a capability change must be treated as a controlled algorithm update. After an algorithm is updated, the agent must register the relevant information: it records the change, the data and task conditions that motivated it, the affected capability version, and the expected performance improvement. It then evaluates the updated capability on a representative test set assembled from accumulated real scheduling instances. The test set should be deliberately designed to cover the relevant task conditions completely and to include difficult, failed, and boundary cases, rather than only the ordinary instances typical of laboratory-level benchmarks. The Assurance Layer provides evidence for these evaluations. The updated capability is accepted for later use when its performance satisfies the predefined criteria and its generated plans pass the Assurance Layer under the represented conditions.

2.3 The Assurance Layer

The Assurance Layer verifies the feasibility of scheduling plans. It not only provides credibility for the Execution Cycle during the execution of each scheduling task, but also serves as the verifier of evaluations during algorithm improvement in the Capability Evolution Layer. It organizes these checks around three functions. Mathematical feasibility checks assess whether a scheduling plan satisfies the constraints represented in the optimization model, with an optimization solver providing the main computational mechanism. Physical consequence checks examine the behavior and consequences of a plan or modification in represented simulation scenarios, with a digital twin serving as a principal implementation. Compliance checks assess whether a scheduling plan conforms to operating and safety requirements that are raised by the industrial site according to specific production conditions and have been explicitly formalized for machine checking, such as maintenance rules or unit operating rules.

3 Three Core Challenges

To make algorithms improve effectively and reasonably

in real industrial environments, a self-evolving optimization agent must address three challenges concerning the amount of data needed for reliable improvement, the alignment of information describing the production system, and the identification of the effects and implications of human actions.

Challenge 1: learning with limited data. Reliable data-driven algorithm improvement requires sufficient data that cover the task conditions in which the algorithm will be used. Industrial scheduling often provides only a limited number of comparable tasks and a small amount of explicit feedback. This makes it difficult to train or update an algorithm from final objective values alone. Lifelong learning and self-evolution methods provide ways to improve optimization strategies over task sequences [5,6], while foundation-model research studies how optimization knowledge may be reused across problem instances [10]. Building on these performance-improvement methods, small-sample learning methods such as meta-learning and transfer learning provide approaches for efficiently improving algorithm performance under limited data.

The central difficulty of data-efficient algorithm improvement in industrial scheduling is that the available cases are few, heterogeneous, and tied to changing physical conditions. Methods that work with limited data must therefore use the structure of the scheduling problem and the information generated during operation. Final objective values and feasibility outcomes are only part of the available evidence. Solver information can reveal active constraints, conflict patterns, search behavior, and recurring sources of infeasibility [11]. Engineer interventions and operating feedback can further indicate why an algorithm's performance changed under a particular condition. These richer sources are often overlooked when learning is organized around the final outcome alone.

The challenge is to combine them with data-efficient learning methods so that a limited number of industrial cases can support reliable improvement in algorithm performance. This challenge directly determines how efficiently the Capability Evolution Layer can update a capability.

Challenge 2: aligning multimodal information with the production system. A scheduling problem draws on different forms of information, such as process flow diagrams, optimization models, equipment data, and connectivity information [12]. These sources describe the same production system from different perspectives. Aligning and integrating information across modalities is the key to preventing the self-evolving system from remaining a purely probabilistic model and instead making it a model consistent with physical mechanisms.

On the one hand, the scarcity of real data is difficult to resolve satisfactorily with generic small-sample learning methods; integrating mechanisms into algorithms through modality alignment can effectively reduce the amount of data required. For example, operating specifications such as unit operating rules and time-interval rules can support the design of better solver heuristics, while unit reaction mechanisms provide the necessary decision basis for diagnosis and adjustment. This avoids learning that depends on large numbers of demonstration labels or trial and error, and improves learning efficiency. On the other hand, the alignment of multimodal information provides the self-evolving system with richer sources of interpretability, which helps it offer engineers more credible, more insightful, and more realistic decision suggestions.

The research challenge is therefore to align heterogeneous information at the critical stages of the workflow. Compared with existing studies on mechanism-informed AI methods, it is necessary, given that scheduling tasks rely heavily on precise numerical computation, to further investigate how multimodal information can be used in numerically sensitive decision-making. This is of significant value both for better Execution Cycle performance and for more sound capability improvement.

Challenge 3: identifying the effects and implications of engineer actions. Engineer actions provide valuable evidence for algorithm improvement, but the meaning of an action may not be clear. Engineers make many kinds of adjustments during scheduling work, such as relaxing a constraint bound, revising a solver setting, or modifying a requirement. The reasons behind these actions may differ: the same action can represent a persistent physical restriction, a temporary production arrangement, a response to a specific risk, or an attempt to improve algorithm performance. Treating engineer actions as imperfect expert demonstrations is more reasonable than taking them as hard rules.

The central challenge is to identify what an action changed, why the change was made, and which consequences followed. Its interpretation requires the agent to connect the action with the task conditions, the preceding plan, the resulting plan, the assurance records, and later operating feedback. Existing human-in-the-loop and continual-learning studies show how human information can be incorporated into adaptive systems [13], but they do not by themselves resolve the industrial problem of attributing the effect of a particular engineer action under changing physical conditions. The research challenge is to extract reliable information about engineer preferences and action effects without treating every local correction as a general rule for future algorithm use.

4 Conclusions

Automatically executing, self-evolving agents are attractive in many domains. Against the background of industrial scheduling tasks, which are safety-critical, complex, and large-scale, this paper proposes a self-evolving system, with the expectation of providing useful discussion for the efficient deployment and effective adaptation of algorithms in real production, thereby improving production efficiency and benefit.

Data availability No datasets were generated or analyzed for this work.

Acknowledgments This work received no specific external funding. The authors used OpenAI Codex solely to improve grammar, clarity, and readability. The authors wrote the manuscript and confirm that its scientific content is original; no AI-generated scientific content, data, or analysis was used. The authors reviewed the language suggestions and take responsibility for the final text.

Competing interests The authors declare no conflicts of interest.

References

1. Belotti P, Kirches C, Leyffer S, et al. Mixed-integer nonlinear optimization. *Acta Numerica*, 2013, 22: 1–131
2. Strang P, Alès Z, Bissuel C, et al. A Markov decision process for variable selection in branch & bound. In: *Proceedings of the 39th International Conference on Neural Information Processing Systems (NeurIPS)*. San Diego, USA, 2025, 3062
3. Strang P, Alès Z, Bissuel C, et al. Planning in branch-and-bound: model-based reinforcement learning for exact combinatorial optimization. In: *Proceedings of the 40th AAAI Conference on Artificial Intelligence*. Singapore, 2026, 25627–25635
4. Liu H Y, Wang J, Geng Z J, et al. Apollo-MILP: an alternating prediction-correction neural solving framework for mixed-integer linear programming. In: *Proceedings of the 13th International Conference on Learning Representations (ICLR)*. Singapore, 2025
5. Pei J Y, Mei Y, Liu J L, et al. Lifelong learning with behavior consolidation for vehicle routing. In: *Proceedings of the 14th International Conference on Learning Representations (ICLR)*. Rio de Janeiro, Brazil, 2026
6. Choi I, Shin W J, Cho S H, et al. Towards generalizable multi-policy optimization with self-evolution for job scheduling. In: *Proceedings of the 39th International Conference on Neural Information Processing Systems (NeurIPS)*. San Diego, USA, 2025
7. Qing H F, Zhang M, Wu Y X, et al. Instance-wise adaptive scheduling via derivative-free meta-learning. In: *Proceedings of the 14th International Conference on Learning Representations (ICLR)*. Rio de Janeiro, Brazil, 2026
8. Yang Q W, Wang L Y, Wicker J, et al. Continual learning: a

- systematic literature review. *Neural Networks*, 2026, 195: 108226
9. Alcobaça E, de Carvalho A C P L F. A literature review on automated machine learning. *Artificial Intelligence Review*, 2026, 59(1): 5
 10. Li S R, Kulkarni J, Menache I, et al. Towards foundation models for mixed integer linear programming. In: *Proceedings of the 13th International Conference on Learning Representations (ICLR)*. Singapore, 2025
 11. Achterberg T. Conflict analysis in mixed integer programming. *Discrete Optimization*, 2007, 4(1): 4–20
 12. Yuan H, Ouyang W L, Zhang C W, et al. OPTFM: a scalable multi-view graph transformer for hierarchical pre-training in combinatorial optimization. In: *Proceedings of the 39th International Conference on Neural Information Processing Systems (NeurIPS)*. San Diego, USA, 2025, 1760
 13. Yang W, Cao D F, Pang J C, et al. Adaptive collaboration with humans: metacognitive policy optimization for multi-agent LLMs with continual learning. In: *Proceedings of the 14th International Conference on Learning Representations (ICLR)*. Rio de Janeiro, Brazil, 2026



operations research, and production planning and scheduling optimization in process industries.

Tianao Kang received the B.S. degree in Automation from Tsinghua University, Beijing, China, in 2023. He is currently pursuing the Ph.D. degree in Control Science and Engineering with the School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China. His current research



applications.

Luolin Xiong received the B.S. and Ph.D. degrees in School of Control Science and Engineering from East China University of Science and Technology, Shanghai, China, in 2020 and 2025, respectively. She is currently a Postdoctoral Researcher at East China University of Science and Technology. Her research interests include smart grid, microgrids, reinforcement learning and their



Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology,

Wei Du (Senior Member, *IEEE*) received the B.S. and M.S. degrees in Electrical Engineering from Donghua University, Shanghai, China, in 2009 and 2012, respectively, and the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, China, in 2016. He is currently a Professor with the Department of Automation and the Key Laboratory of Smart

Shanghai, China. His current research interests include machine learning-assisted evolutionary optimization, large-scale optimization, robust optimization, and decision-making optimization of complex industrial process.



Yang Tang (Fellow, *IEEE*) received the B.S. and Ph.D. degrees in Electrical Engineering from Donghua University, Shanghai, China, in 2006 and 2010, respectively. From 2008 to 2010, he was a Research Associate with The Hong Kong Polytechnic University, Hong Kong, China. From 2011 to 2015, he was a Postdoctoral Researcher with The Humboldt University of Berlin, Berlin, Germany, and with the Potsdam Institute for Climate Impact Research, Potsdam, Germany. He is now a Professor with the East China University of Science and Technology, Shanghai. His current research interests include distributed estimation/control/optimization, agentic AI, computer vision, reinforcement learning, cyber-physical systems, hybrid dynamical systems, and their applications. Prof. Tang is an *IEEE* Fellow. He was a recipient of the Alexander von Humboldt Fellowship. He is a Co-Editor-in-Chief of *IEEE Transactions on Industrial Informatics* (and was an Associate Editor), a Senior Area Editor/Associate Editor of *IEEE Transactions on Circuits and Systems-I: Regular Papers*, Associate Editor of *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cybernetics*, *IEEE/ASME Transactions on Mechatronics*, *IEEE Transactions on Cognitive and Developmental Systems*, *IEEE Transactions on Emerging Topics in Computational Intelligence*, *IEEE Systems Journal*, *Engineering Applications of Artificial Intelligence (IFAC Journal)*, *Science China Information Sciences*, *Science China Technological Sciences* and *Acta Automatica Sinica*, etc. He is an *IEEE* Distinguished Lecturer from 2026–2027. He has published more than 200 papers in international journals and conferences, including more than 160 papers in *IEEE Transactions* and 20 papers in *IFAC* journals. He has been awarded as best/outstanding Associate Editor in *IEEE* journals for seven times. He is a (leading) guest editor for several special issues focusing on autonomous systems, robotics, and industrial intelligence in *IEEE Transactions*.



Feng Qian (Senior Member, *IEEE*) received the B.S. degree in Chemical Automation and Meters from the Nanjing Institute of Chemical Technology, Nanjing, China, in 1982, and the M.S. and Ph.D. degrees in Automation from the East China Institute of Chemical Technology, Shanghai, China, in 1988 and 1995, respectively. He was the Director of the Automation Institute, East China University of Science and Technology, from 1999 to 2001, and was the Head of the Scientific and Technical Department from 2001 to 2006. He is currently the Vice President of the East China University of Science and Technology, the Director of the Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, Shanghai, and the Director of the Process System Engineering Research Center, Ministry of Education, Shanghai. His current research interests include modeling, control,

optimization, and integration of petrochemical complex industrial processes and their industrial applications, neural network theory, and real-time intelligent control technology and its applications to the ethylene, PTA, PET, and refining industries. Prof. Qian is a

member of the China Instrument and Control Society, the Chinese Association of Higher Education, and China's PTA Industry Association. He is also an Academician of the Chinese Academy of Engineering.