

Article

A Dual-Stream Deep Reinforcement Learning Framework for Hot Rolling Production Scheduling

Chi Wang, Wang Cao * and Min Huang

College of Information Science and Engineering, Northeastern University, Shenyang 110819, China; 20235483@stu.neu.edu.cn (C.W.); mhuang@mail.neu.edu.cn (M.H.)

* Correspondence: 2310265@stu.neu.edu.cn

Abstract

Hot Rolling Production Scheduling (HRPS) is a crucial combinatorial optimization problem characterized by severe conflicts between rigid physical rolling rules and strict order due dates. While real-time scheduling is essential for dynamic manufacturing, traditional meta-heuristics suffer from severe computational time bottlenecks. Conversely, standard end-to-end Deep Reinforcement Learning (DRL) models offer rapid inference but typically struggle with spatio-temporal feature entanglement, training instability under extreme penalty landscapes, and poor zero-shot scale generalization. To bridge these gaps, this paper proposes a novel framework named Dual-Stream Group-Optimize Policy Optimization with Multiple Optima (DSGO-POMO). The framework introduces three core innovations: (1) a Dual-Stream intervention network that explicitly decouples and synergistically fuses physical attributes with temporal slacks; (2) a Group Relative Policy Optimization (GRPO) training mechanism to stabilize policy updates; and (3) an Entropy-Aware and Dual-Annealed Differential Active Search (EA-DAS) strategy to seamlessly adapt pre-trained weights to out-of-distribution scales. Extensive computational experiments validate the superiority of the proposed framework. On medium-scale instances ($n = 50$), the basic DSGO-POMO slashes inference time to merely 0.09 min compared to the approximately 46 min required by classical heuristics while simultaneously achieving lower total costs. Furthermore, during zero-shot extrapolation on ultra-large instances ($n = 200$), the framework coupled with EA-DAS outperforms the best heuristic baseline by 14.7% in solution quality and completes the search in under 2 min. These numerical breakthroughs confirm that DSGO-POMO provides a blazing-fast, highly robust, and industrially viable rescheduling paradigm.

Keywords: hot rolling production scheduling; spatio-temporal decoupling; group relative policy optimization; active search; combinatorial optimization

Academic Editor: Alexios Papacharalampopoulos

Received: 15 May 2026

Revised: 25 June 2026

Accepted: 26 June 2026

Published: 30 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Hot Rolling Production Scheduling (HRPS) links the upstream continuous casting stage with downstream cold rolling operations. Planners must extract slabs from a large order pool and sequence them into rolling units based on strict process rules. Unlike basic sequencing tasks, HRPS forces a direct trade-off between physical machinery limits and production deadlines. On the shop floor, the sequence usually needs to follow a “coffin-shaped” width profile to control thermal shock and roll wear. At the same time, the

schedule must keep tardiness penalties as low as possible. This tight coupling of physical restrictions and temporal urgency makes HRPS an NP-hard combinatorial optimization problem [1].

Building upon this established complexity, past research has often treated HRPS as a highly constrained prize-collecting vehicle routing problem (PCVRP) and solved it using various meta-heuristics [1,2]. These methods work reasonably well for small, static scheduling instances. However, they hit severe computational bottlenecks when the order pool grows or the shop floor faces unexpected changes. A larger number of slabs leads to a combinatorial explosion. Furthermore, during real-time rescheduling—such as when equipment fails or a rush order arrives—traditional heuristics are simply too slow to generate feasible, high-quality sequences within the required time window.

To bypass the speed limits of iterative search, the field is increasingly exploring end-to-end learning methods. Deep Reinforcement Learning (DRL) has emerged as a strong alternative for combinatorial optimization, with attention-based policy networks showing extremely fast inference times on standard routing tasks like the TSP and VRP. However, a critical research gap exists in extending these successes to highly constrained industrial processes like HRPS. Hot rolling involves much stricter physical limits (e.g., the non-increasing width rule), non-negotiable delivery dates, and an uneven penalty landscape. Consequently, standard policy learning methods typically struggle with three critical issues: spatio-temporal feature entanglement, gradient instability caused by high-variance penalties, and severe generalization degradation when tested on out-of-distribution (OOD) scales. Essentially, existing DRL approaches for complex scheduling primarily treat the task as a generic black-box optimization problem, often overlooking the rigid, asymmetric metallurgical constraints inherent in the process. Such data-driven methods frequently struggle with constraint violation and require extensive hyperparameter tuning to converge. This study diverges from these paradigms by introducing a ‘physically-informed’ scheduling framework. Unlike existing methods that rely on single-stream architectures—which often lead to spatio-temporal feature entanglement—the proposed framework explicitly embeds the physical semantics of the HRPS process (such as the non-increasing width rule) directly into the decision-making pipeline. This conceptual shift from purely data-driven to physically informed optimization is the fundamental distinction between this research and existing DRL baselines.

To bridge the aforementioned gaps, this article formulates the sequence generation process of HRPS as a Markov Decision Process and proposes DSGO-POMO, a novel Dual-Stream Group-Optimize POMO framework. The main contributions are three-fold: First, a Dual-Stream intervention network is developed to decouple physical routing constraints from temporal urgency, successfully preventing spatio-temporal feature entanglement. Second, a POMO-enhanced Group Relative Policy Optimization (GRPO) mechanism and an EA-DAS active search strategy are designed to respectively stabilize offline training under extreme penalty landscapes and enhance zero-shot scale generalization. Finally, extensive experiments confirm that the proposed framework significantly outperforms traditional heuristics and standard DRL baselines in both computational efficiency and solving quality, providing a blazing-fast and highly robust rescheduling paradigm for dynamic industrial cyber-physical systems.

The remainder of this article is organized as follows: Section 2 reviews the related literature. Section 3 presents the mathematical model of the HRPS problem. Section 4 details the DSGO-POMO architecture, the GRPO training mechanism, and the EA-DAS strategy. Section 5 presents and analyzes the experimental results. Finally, Section 6 concludes this article and outlines future research directions.

2. Literature Review

The HRPS problem represents one of the most highly constrained and economically critical processes in modern metallurgical cyber–physical systems. To contextualize the architectural contributions within the broader landscape of intelligent scheduling, the literature is reviewed across three pivotal domains, tracing the evolution from conventional operations research (OR) to modern learning-based paradigms.

2.1. Exact Methods and Meta-Heuristics

Fundamentally, the core objective of HRPS is to extract and sequence optimal slabs from a candidate pool, maximizing economic yield while satisfying rigorous cyber–physical constraints. In early mathematical modeling, HRPS was frequently abstracted as a prize-collecting traveling salesman problem (PCTSP) [1,2]. However, constrained by its single-tour topology, the PCTSP paradigm structurally fails to partition multiple rolling units or accommodate complex physical transition rules. To overcome this topological limitation, subsequent OR studies upgraded HRPS modeling to highly constrained variants of the prize-collecting vehicle routing problem (PCVRP). Under this advanced framework, exact mathematical programming and decomposition strategies have been rigorously explored. Notably, Mao et al. [3] thoroughly elucidated a Lagrangian relaxation approach using a deflected conditional subgradient method to decouple complex machine capacity constraints. Similarly, hybrid approaches utilizing mixed-integer programming (MIP) and constraint programming (CP) via logic-based Benders decomposition [4] were proposed to tackle coupled multistage scheduling.

While yielding mathematically rigorous bounds, these exact methods suffer from prohibitive computational latency. Driven by this bottleneck, industrial applications predominantly shifted to meta-heuristic solvers. Tang et al. [5] applied an iterated local search (ILS), while Chen et al. [6] proposed a hybrid evolutionary strategy. For multi-objective models, Jia et al. [7] introduced a Pareto-dominance-based ant colony system (ACS). Furthermore, diverse swarm intelligence and neighborhood search algorithms have been adapted for complex metallurgical scheduling. For instance, Tang and Wang [8] combined discrete particle swarm optimization (PSO) with local search, Jiang et al. [9] utilized a two-phase soft optimization method to handle processing time uncertainties, and Tang et al. [10] applied a scatter search algorithm incorporating path-relinking to address severe blocking constraints.

Responding to escalating industrial uncertainties, recent studies evolved towards robust and dynamic scheduling paradigms [11–13]. Pushing the boundaries of adaptability, Jiang et al. [14] integrated Q-learning into an evolutionary framework for adaptive operator selection. Nevertheless, despite their empirical success, these iterative search algorithms inherently rely on time-consuming generation-and-evaluation cycles from scratch. This structural lack of autonomous learning capabilities and millisecond-level responsiveness renders them intractable for real-time cyber–physical control, catalyzing the urgent shift toward end-to-end deep reinforcement learning (DRL) paradigms.

2.2. Deep Reinforcement Learning for Complex Scheduling

Neural combinatorial optimization (NCO) using deep reinforcement learning (DRL) directly addresses the latency issues of exact solvers and heuristics. Because these models learn problem distributions offline, they can generate schedules in milliseconds during testing. Early work relied on Sequence-to-Sequence models and Pointer Networks [15]. However, supervised learning requires expensive optimal labels, making it impractical for large industrial tasks. Bello et al. [16] removed this data barrier by introducing the actor–critic framework and REINFORCE algorithm for purely unsupervised training.

Later, Kool et al. [17] built the Attention Model (AM). Their use of a Transformer-based self-attention mechanism significantly improved routing performance.

A major drawback of standard AMs is their reliance on simple greedy or beam search decoding. This causes them to easily get stuck in local optima when the solution space is large. Kwon et al. [18] solved this by developing the policy optimization with multiple optima (POMO) framework. POMO forces the model to start from several distinct nodes and uses their average reward as a shared baseline. This strategy drops the need for a Critic network and drastically reduces local optima traps. Researchers quickly adapted these DRL foundations to shop scheduling. Song et al. [19] combined graph representations with DRL for flexible job shops, while Lei et al. [20] and Luo [21] handled dynamic job insertions using multi-action Markov decision models. DRL eventually moved into tougher continuous manufacturing scenarios. Pan et al. [22] applied it to steelmaking-continuous casting, and Kong et al. [23] used DQN for batch processing in semiconductor plants.

Even with these advances, applying general NCO frameworks directly to HRPS does not work well. First, HRPS has strict asymmetric constraints, such as the “non-increasing width” rule. This physical restriction instantly breaks POMO’s uniform multi-start mechanism because picking a narrow slab first creates permanent deadlocks. Second, the problem contains heavily coupled constraints. To fix the weaknesses of flat sequence processing in Transformers, studies have turned to Graph Neural Networks (GNNs) to explicitly model topologies [24–26]. Xu et al. [27] used multiple relational attention for dynamic contexts, and Zhu et al. [28] applied heterogeneous GNNs to separate coupled resources. Even the latest DRL model specifically designed for hot rolling [29] still relies on a standard single-stream network to map all slab features into one latent space. A structured comparison of these representative recent reinforcement learning-based scheduling studies is provided in Table 1, highlighting the methodological evolution and the limitations of existing approaches.

Table 1. Overview of methodological paradigms and application characteristics in recent RL-based scheduling studies.

Ref.	Year	Methodology	Focus Area	Constraint Handling
Kool et al. [17]	2018	Attention Model	Routing	General sequence
Kwon et al. [18]	2020	POMO	Routing	Multi-optima
Song et al. [19]	2022	GNN + DRL	Job-Shop	Graph-based
Lei et al. [20]	2022	Multi-Action DRL	Job-Shop	Dynamic insertion
Pan et al. [22]	2024	DRL	Casting	Time-of-use tariffs
Kong et al. [23]	2024	DQN	Batch	Energy-efficient
Che et al. [29]	2026	DRL	Hot Rolling	Energy-efficient

Building upon the aforementioned graph-based approaches, the widespread deployment of Industrial Internet of Things (IIoT) sensors has introduced new dimensions to intelligent manufacturing. Recent studies emphasize the role of cognitive sensing and IoT-based optimization in bridging the gap between shop-floor data and algorithmic decision-making. For instance, Su et al. [30] proposed a heterogeneous graph neural network to mine spatial-temporal relations across manufacturing elements, enabling robust product-level energy prediction. Similarly, Akashwar et al. [31] demonstrated that integrating machine learning forecasts with real-time IoT data acquisition significantly improves predictive load scheduling in decentralized energy systems.

Driven by these rich data pipelines, recent AI advancements have further diversified the architectural paradigms for combinatorial optimization. For instance, Multi-Agent Reinforcement Learning (MARL) has emerged as a robust framework for complex system

coordination, often utilizing Centralized Training with Decentralized Execution (CTDE) to optimize coupled multi-stage processes. Meanwhile, Transformer-based architectures, specifically those leveraging Decision Transformer principles and efficient attention mechanisms, have revolutionized sequence modeling by enabling agents to extract deeply coupled, long-range dependencies across massive action spaces without the gradient instability typical of earlier RNNs. Recognizing that HRPS fundamentally requires rigorously evaluating sequence-dependent dimensional transitions over long time horizons under rigid physical constraints, this study deliberately adopts a Transformer-based architecture.

2.3. Policy Optimization Dynamics and Inference-Time Search

Although POMO-based architectures perform well in standard routing tasks, their training still depends on the basic REINFORCE algorithm. Applying this simple policy gradient method directly to the strict constraints of HRPS exposes critical flaws. HRPS involves non-linear continuous temporal dynamics, stringent penalty terms, and asymmetric transition costs. In such a high-constraint landscape, the standard REINFORCE algorithm lacks a fine-grained relative advantage measure among the population, inducing extreme gradient variance. Although advanced trust-region and actor–critic architectures, such as PPO [20,21] and advantage actor–critic [32], have been introduced to mitigate such variance in general scheduling tasks, they encounter distinct computational bottlenecks in HRPS. The massive, dynamically masked combinatorial action spaces of HRPS render the training of their auxiliary value networks highly unstable and computationally prohibitive. More crucially, these standard actor–critic paradigms evaluate actions against an absolute state-value baseline, structurally failing to capture the fine-grained relative advantages among a parallel solution population.

Furthermore, the inherently low sample efficiency of these on-policy algorithms—where interaction data is discarded immediately after a single gradient update—poses a significant challenge given the prohibitive simulation costs of HRPS. Consequently, high-penalty exploratory trajectories frequently trigger invalid actions, leading to collapsed reward signals and trapping the policy in sub-optimal local minima. To circumvent this generalization ceiling and break the solving upper limit on highly constrained instances, Bello et al. [16] initially explored Active Search during the testing phase. To reconcile the prohibitive computational latency of standard AS, Hottung et al. [33] introduced EAS to update only a lightweight subset of parameters. Crucially, without concurrently resolving the underlying gradient variance and feature entanglement during the offline training phase, relying solely on test-time updates to efficiently navigate the vast HRPS solution space remains a sub-optimal and computationally demanding task.

2.4. Research Gaps and Proposed Paradigm

Synthesizing the preceding analyses of policy optimization dynamics, a systematic examination of the current literature reveals three major bottlenecks in HRPS: (1) the computational delay of traditional heuristics; (2) the feature entanglement in existing neural architectures when handling spatio-temporal conflicts; and (3) the gradient instability under extreme penalties, along with the need for more efficient test-time fine-tuning. To bridge these gaps, the DSGO-POMO framework is proposed. This framework explicitly separates physical and temporal features using a Dual-Stream network and stabilizes offline training via Group Relative Policy Optimization (GRPO). Furthermore, an EA-DAS strategy is applied during inference. Because the pre-trained model already exhibits strong zero-shot generalization, EA-DAS acts as a targeted refinement mechanism to further optimize the sequences and moderately reduce costs, rather than merely rescuing the

model from catastrophic degradation. Together, these mechanisms provide a fast, stable, and highly competitive solver for dynamic industrial scheduling.

3. Problem Formulation

3.1. Problem Description

As illustrated in Figure 1, the HRPS problem requires selecting appropriate slabs from a dynamic order pool and sequencing them into contiguous rolling units under stringent mechanical and thermodynamic constraints. A complete rolling unit comprises a short “warming up” phase and a primary “coming down” phase. Because the auxiliary slabs for warming up are limited and typically pre-assigned, the scheduling optimization in this study focuses exclusively on the highly combinatorial “coming down” phase. During this phase, to prevent irreversible localized wear and catastrophic thermal shock to the roll surfaces, the scheduled sequence must strictly adhere to a “non-increasing width” rule, thereby naturally forming a “coffin-shaped” profile. Additionally, the cumulative length of consecutively rolled slabs with identical widths is restricted by a localized wear limit, and the total operational length of a single rolling unit is absolutely bounded by the roll’s physical fatigue limit.

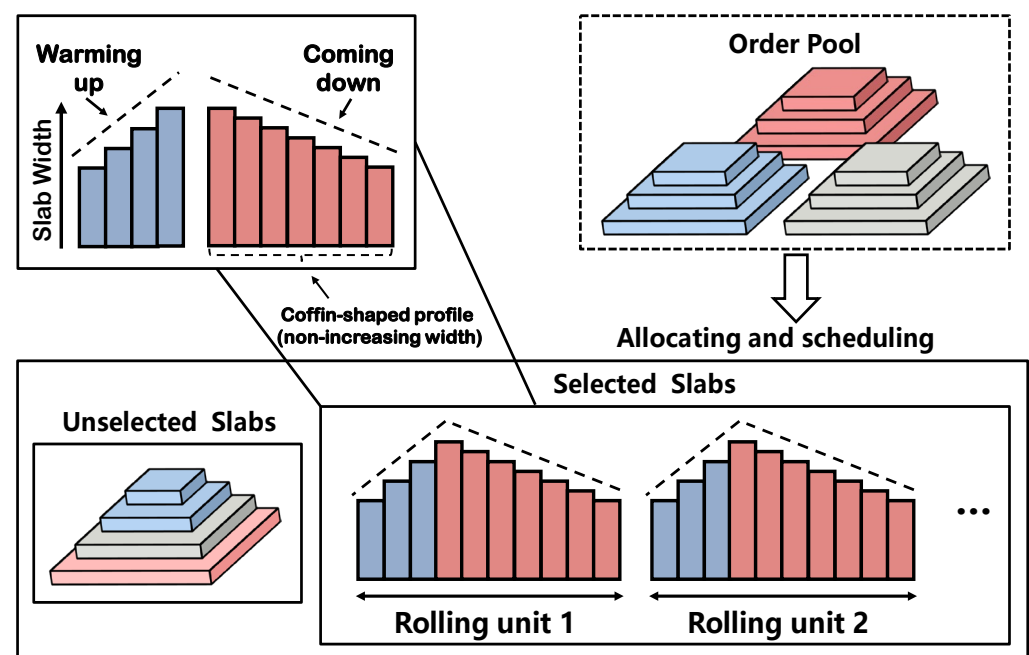


Figure 1. The physical process and schematic diagram of HRPS. In the diagram, the blue and red blocks represent slabs assigned to the ‘warming up’ and ‘coming down’ phases, respectively. The dashed lines illustrate the ideal coffin-shaped width profile. The dashed frame indicates the in-ital candidate order pool, whereas the solid frames enclose and categorize the scheduling results, such as the se-lected rolling units and the unselected slabs.

Beyond satisfying these rigid physical thresholds, an optimal HRPS sequence must achieve a delicate trade-off among conflicting economic and temporal indicators. The primary objective of the scheduling system is to minimize the total comprehensive production cost. This encompasses the fixed capital cost of initiating new rolls, the transition penalties induced by abrupt variations in physical dimensions (width, thickness, and hardness) between adjacent slabs, and the default penalties for unselected orders. Fundamentally, this highly constrained industrial process is mathematically abstracted as a variant of the Prize-Collecting Vehicle Routing Problem with Time Windows (PCVRPTW). By

assuming a constant unit rolling speed, the complex thermodynamic sequencing task is rigorously transformed into a discrete routing model. This assumption eliminates stochastic processing fluctuations, thereby isolating the impact of combinatorial complexity and establishing a theoretical optimality baseline for evaluating the scheduling sequences.

3.2. Model Formulation

To rigorously evaluate the HRPS problem quantitatively and establish a theoretical foundation for the subsequent Markov Decision Process formulation, this section mathematically models the problem as a Prize-Collecting Vehicle Routing Problem with Time Windows variant subject to complex physical constraints.

3.2.1. Sets and Indices

Assumption 1 (Constant Rolling Speed and Time Mapping). *The rolling speed of the hot strip mill is assumed to be constant during the processing of a single rolling unit. Consequently, the processing time (p_i) of any slab is directly proportional and numerically equivalent to its physical processing length (l_i).*

Assumption 2 (Deterministic Attributes). *Unlike robust optimization models that handle uncertain processing disturbances, this study leverages the millisecond-level inference capability of DRL to execute real-time deterministic rescheduling. Therefore, within a single decision horizon, the physical attributes (width, thickness, hardness) and due dates of all candidate slabs are assumed to be deterministic and known a priori.*

3.3. Objective Function and Constraints

$$\min Z = \sum_{j \in N} C_0 x_{0,j} + \sum_{i \in N_+} \sum_{j \in N_+} c_{i,j} x_{i,j} + \sum_{i \in N} p v_i (1 - y_i), \quad (1)$$

$$\text{s.t. } \sum_{j \in N_+, j \neq i} x_{i,j} = y_i, \forall i \in N, \quad (2)$$

$$\sum_{i \in N_+, i \neq j} x_{i,j} = y_j, \forall j \in N, \quad (3)$$

$$w_j \leq w_i + M(1 - x_{i,j}), \forall i \in N_+, j \in N, \quad (4)$$

$$cl_j \geq cl_i + l_j - M(1 - x_{i,j}) - MI(w_i \neq w_j), \forall i, j \in N, \quad (5)$$

$$cl_j \leq L^w, \forall j \in N, \quad (6)$$

$$u_j \geq u_i + l_j - M(1 - x_{i,j}), \forall i \in N_+, j \in N, \quad (7)$$

$$u_j \leq L_{\max}, \forall j \in N, \quad (8)$$

$$u_i \geq L_{\min} x_{i,0}, \forall i \in N, \quad (9)$$

$$\tau_i + p_i \leq \tau_j + M(1 - x_{i,j}), \forall i, j \in N, \quad (10)$$

$$\tau_i + p_i + T_{\text{setup}} \leq \tau_j + M(2 - x_{i,0} - x_{0,j}), \forall i, j \in N, \quad (11)$$

$$e_i y_i \leq \tau_i, \forall i \in N, \quad (12)$$

$$\tau_i + p_i \leq d_i + M(1 - y_i), \forall i \in N, \quad (13)$$

$$x_{i,j} \in \{0,1\}, \forall i, j \in N_+, \quad (14)$$

$$y_i \in \{0,1\}, \forall i \in N, \quad (15)$$

$$u_i, cl_i, \tau_i \geq 0, \forall i \in N_+, \quad (16)$$

The objective function defined in Equation (1) aims to minimize the total production cost, which is formulated as the sum of three components: the fixed roll-changing cost incurred by initiating new rolling units, the transition penalty cost arising from physical attribute disparities between adjacent slabs, and the default penalty cost induced by unselected slabs retained in the candidate pool.

Equations (2) and (3) establish the network flow balance of the scheduling process. They mandate that any physical slab incorporated into the production plan must possess exactly one predecessor and one successor. Conversely, if a slab is unselected, its corresponding routing paths are forcibly severed. Equation (4) enforces the paramount non-increasing width rule characteristic of HRPS, strictly ensuring that the width of any subsequent slab does not exceed that of its immediate predecessor within the same rolling unit.

Equations (5) and (6) track and restrict the continuous rolling length for slabs of identical widths. Specifically, in Equation (5), $I(\cdot)$ is an indicator function that equals 1 if the condition inside is true (i.e., adjacent slabs have different widths, $w_i \neq w_j$), and 0 otherwise. Driven by this indicator and the Big-M parameter, the equation ensures that the accumulated local wear length is continuously recorded only when adjacent slabs share the same width. Subsequently, this accumulated length must not surpass the safety threshold L_w , which is strictly enforced by Equation (6).

Equation (7) monitors the cumulative processing length of the entire rolling unit, effectively eliminating illegal disjoint sub-tours mathematically. Equations (8) and (9) bound the capacity of a single rolling unit. They guarantee that the cumulative processing length remains below the absolute fatigue limit L_{max} , while satisfying the minimum economic threshold L_{min} prior to executing a roll-changing operation.

Equations (10)–(13) establish the temporal deduction trajectory and hard time-window boundaries. Equations (10) and (11) track the production timeline. Equation (10) links the start times of adjacent slabs in the same unit, and Equation (11) adds the required setup time when starting a new roll. Equation (12) prevents any selected slab from starting before its earliest release time. Similarly, Equation (13) forces the completion time to meet the required due date. The Big-M formulation is employed to automatically drop these time limits for any unselected orders. Lastly, Equations (14)–(16) set the variable types and non-negativity rules.

In summary, the mathematical formulations defined in this section provide the rigorous boundary conditions that govern the reinforcement learning environment. The comprehensive cost function expressed in Equation (1) is directly transformed into the global delayed reward signal to guide policy optimization, while the rigid physical constraints—specifically the width profile and roll capacity rules described in Equations (4)–(9) form the logic of the dynamic masking mechanism to ensure sequence feasibility. Furthermore, the temporal deduction and hard due-date boundaries established in Equations (10)–(13) provide the mathematical derivation for calculating temporal slacks, which serve as the core input features for the Dynamic Temporal Intervention Stream within our proposed dual-stream architecture. By systematically mapping these Mixed-Integer Programming rules into the deep reinforcement learning pipeline, the agent is able to internalize both the metallurgical logic and the production urgency requirements during its self-play exploration, effectively bridging the gap between the static mathematical model and the dynamic scheduling framework.

4. Dual-Stream Group-Optimize POMO

To address the limitations of existing DRL frameworks, a Dual-Stream architecture is proposed to explicitly address the challenge of feature entanglement. In traditional DRL scheduling models, spatial physical constraints (e.g., transition rules) and temporal objective functions (e.g., cost minimization) are processed within a unified, entangled latent space, inevitably leading to gradient interference and optimization instability. The proposed framework explicitly decouples these components: the spatial stream is architecturally restricted to ensure constraint legality, while the temporal stream functions as an adaptive urgency amplifier. By structurally segregating these fundamentally conflicting optimization goals, this approach achieves a level of stability and constraint-adherence that standard single-stream architectures cannot sustain.

4.1. Markov Decision Process Modeling from a Sequence Generation Perspective

Distinct from conventional exact or heuristic algorithms that rely on branch-and-bound or local search, DRL implicitly learns the global probability distribution of node sequences through environment interactions. To efficiently solve the static PCVRPTW-variant model formulated in Section 3 using an autoregressive network, this section first reformulates HRPS as a dynamic Markov Decision Process. In this formulation, the scheduling scheme is conceptualized as a sequence generation task: at each discrete time step t , based on the current shop-floor state, the agent sequentially selects the next node π_t from the candidate slab pool or the dummy start node, until all schedulable slabs are allocated or the network's termination condition is triggered.

4.1.1. State Space

At any discrete time step t , the system environment state s_t observed by the agent must comprehensively capture both the global static attributes and the local dynamic evolution process. This comprehensive representation assists the agent in achieving an optimal trade-off between mechanical physical constraints and due-date default costs. The state s_t is jointly composed of static graph node features and dynamic context states.

Static Graph Node Features: This component encompasses the inherent physical dimensions and business attributes of all elements within the extended node set N^+ . To accelerate the gradient convergence of the policy network, rigorous scaling transformations are applied to all raw physical quantities at the input layer. The normalized node feature vector is explicitly defined as x_i .

Dynamic Context State: To provide high-dimensional decision-making criteria for the autoregressive decoder, the dynamic context vector at time step t is defined as c_t . This context not only embeds the representation of the node scheduled in the previous step but also tracks and updates the following core state variables in real-time across macroscopic and microscopic dimensions:

Macroscopic statistical features of the candidate pool (F_{pool}): Includes the dynamic mean μ and maximum absolute deviation σ of all currently unscheduled slabs across core dimensions (width, thickness, hardness, and default penalties). This feature aims to provide the policy network with a global statistical perception regarding the deviation degree of the remaining order distribution.

Cumulative total length of a single roll (L_{total}): Represents the cumulative processing length of the current rolling unit, utilized to quantify the consumption process of the single roll's fatigue life. **Cumulative length of identical width (L_{width}):** Indicates the cumulative length of consecutively processed slabs with identical widths, used to monitor the localized wear state of the roll. **Cumulative absolute processing time (T_{abs}):** Represents the

cumulative absolute processing time of the current system, establishing a global scheduling timeline for subsequent orders.

Global and individual temporal slack (S_{global} and s_i): To precisely quantify due-date urgency, the absolute temporal slack for any unscheduled slab i in the candidate pool is strictly defined as the difference between its due date and its projected earliest completion time, i.e., $s_i = d_i - \max(e_i, T_{\text{abs}}) - p_i$. The vector S_{global} , composed of the s_i values for all unscheduled slabs, reveals the real-time allowance of each order to the policy network, whereas $\min(S_{\text{global}})$ extracts the current tightest global allowance lower bound. This dual-slack design compels the agent to acutely perceive the default risks of high-priority orders at both microscopic and macroscopic levels.

Before entering the neural network, all continuous state variables must be scaled to a unified range. They are divided by their respective physical upper bounds or estimated maximum completion times. To further stabilize training, any extreme values that might otherwise trigger gradient spikes are clipped.

4.1.2. Action Space

At any discrete time step t , the agent's action a_t is strictly defined as assigning the next node to be scheduled from the extended candidate set N^+ , denoted as $\pi_t = a_t$. This discrete action space encompasses two fundamentally distinct scheduling decisions:

When the action value $a_t \in N$, it signifies that the system decides to extract a physical slab a_t from the candidate pool and append it to the tail of the current rolling sequence.

When the action value $a_t = 0$, it indicates that the system actively terminates the current rolling unit and triggers a physical roll-changing operation. Under this state transition logic, the cumulative total roll length u_t and the cumulative identical-width length cl_t within the dynamic context are both reset to zero. Furthermore, the non-increasing width hard constraint for the first slab of the new roll is automatically relaxed, and the system's cumulative absolute processing time τ_t is synchronously incremented by the fixed roll-changing time overhead.

4.1.3. State Transition and Dynamic Masking Mechanism

To ensure that the generated scheduling sequences strictly satisfy the process and capacity limitations of HRPS, a dynamic masking mechanism is introduced at the action decoding layer. The underlying logic of this mechanism is built upon the forward transition of the environment state.

State Transition Equations: The dynamic context of the environment evolves step-by-step. Rather than relying on arbitrary empirical assumptions, the state transition functions governing these core variables are directly mathematically deduced from the physical constraints formulated in the established Mixed-Integer Programming (MIP) model (Section 3.3). Specifically, the state transitions for the cumulative roll length (u_t), the cumulative identical-width length (cl_t), and the cumulative absolute processing time (τ_t) act as the discrete Markovian equivalents of Equations (7), (5), (10) and (11), respectively. They are strictly defined as follows:

$$u_{t+1} = \begin{cases} 0, & \text{if } a_t = 0 \\ u_t + l_{a_t}, & \text{if } a_t \in N' \end{cases} \quad (17)$$

$$cl_{t+1} = \begin{cases} 0, & \text{if } a_t = 0 \\ cl_t + l_{a_t}, & \text{if } a_t \in N \text{ and } w_{a_t} = w_{\pi_{t-1}}, \\ l_{a_t}, & \text{if } a_t \in N \text{ and } w_{a_t} < w_{\pi_{t-1}} \end{cases} \quad (18)$$

$$\tau_{t+1} = \begin{cases} \tau_t + T_{\text{setup}}, & \text{if } a_t = 0 \\ \tau_t + p_{a_t}, & \text{if } a_t \in N' \end{cases} \quad (19)$$

As established in the mathematical formulation, T_{setup} denotes the fixed time parameter required to initiate a new rolling unit. Governed by this deterministic transition logic derived from the MIP model, the physical attributes (such as the temporal slack) undergo strict synchronous updates, ensuring that the generated sequence natively satisfies all physical boundaries.

Dynamic Masking Mechanism: Based on the aforementioned state transition rules, the system constructs a dynamic mask set $\mathcal{M}_t$ in real-time at each time step to screen out the infeasible search space. For any extended candidate node $j \in N^+$, if it triggers any of the following conditions, the system forcibly assigns it to the infeasible mask set $\mathcal{M}_t$:

Repeated visit restriction: The system automatically masks all physical slab nodes that have already been scheduled in previous steps. **Non-increasing width blockade:** If the width of the candidate slab is strictly greater than the width of the preceding slab on the current roll, the action is blocked. This constraint is automatically relaxed when the preceding node is the dummy start node. **Fatigue life overflow blockade:** If the estimated cumulative roll length after appending the candidate slab exceeds the physical fatigue limit, the node is masked.

Localized excessive wear blockade: Under the premise that the slab width does not decrease, if the estimated continuous processing length of identical width surpasses the safety threshold, this option is discarded. **Inevitable tardiness truncation:** If the estimated completion time of a candidate slab strictly exceeds its designated due date, resulting in inevitable default, the system directly blocks this scheduling option to avert meaningless delayed scheduling. **Roll-changing economic lower bound:** The system permits the execution of a roll-changing action if and only if the cumulative processing length of the current roll reaches or exceeds the minimum economic production threshold. If $u_t < L_{min}$, the action pointing to the dummy node is forcibly masked.

Adaptive deadlock termination: A deadlock occurs if physical conflicts mask all remaining slabs before the cumulative roll length reaches L_{min} . When this happens, the environment ends the episode early. To prevent gradient explosions, artificially huge penalties are not applied. Instead, the system simply calculates the actual tardiness costs for all unscheduled slabs. This approach keeps the training numerically stable while using natural negative feedback to steer the network away from dead-end states.

Start node exploration filtering: Because each roll must follow the non-increasing width rule, starting a new roll with a narrow slab wastes search space. Therefore, when opening a new roll, the system uses width quantiles to mask the narrowest candidate slabs. This simple pruning step cuts off useless search branches early and makes the multiple-optima search much more efficient.

Finally, before calculating the action probabilities, the system sets the predicted logits of all masked nodes in $\mathcal{M}_t$ to negative infinity. After the Softmax layer, their selection probabilities become exactly zero. This direct masking process ensures that the network strictly obeys all physical hard constraints, successfully replacing the complex Big-M inequalities used in exact solvers.

4.1.4. Reward Function

The optimization objective of reinforcement learning is to maximize the cumulative return. To maintain strict consistency with the mathematical model established in Section 3, this article adopts a global delayed reward mechanism. Upon the complete generation of a scheduling sequence π or the triggering of an early termination condition, the environment provides a one-time scalar reward $R(\pi)$ for that trajectory.

The reward function is defined directly as the negative value of the actual total cost incurred in the objective formula (1), where the three respective terms correspond to the

fixed roll-changing cost of initiating new rolling units, the physical transition penalty between adjacent slabs, and the due-date default penalty of unscheduled slabs.

$$R(\pi) = -\left(\sum_{t=0}^{T-1} C_0 I(\pi_t = 0) + \sum_{t=0}^{T-1} c_{\pi_t, \pi_{t+1}} + \sum_{i \in N \setminus \pi} p v_i\right), \quad (20)$$

4.2. Architecture of DSGO-POMO

When solving combinatorial optimization problems like HRPS, which entail massive discrete solution spaces and stringent physical process constraints, conventional autoregressive DRL models typically generate only a single rollout trajectory during forward propagation. This limitation makes them highly susceptible to falling into sub-optimal traps or even experiencing deadlocks within a search space fraught with hard constraints. To break through this bottleneck, this article adopts the Policy Optimization with POMO framework as the core algorithmic foundation.

The foundation of this policy network architecture discards traditional RNNs in favor of an encoder-decoder topology based entirely on the self-attention mechanism. Crucially, to accommodate the core philosophy of POMO, the tensor data flow of the standard attention network is profoundly reconstructed tailored to the dynamic evolution characteristics of HRPS. This modification enables native support for low-level tensor parallel inference of “single-instance, multiple-trajectories.” The overall network architecture is illustrated in Figure 2.

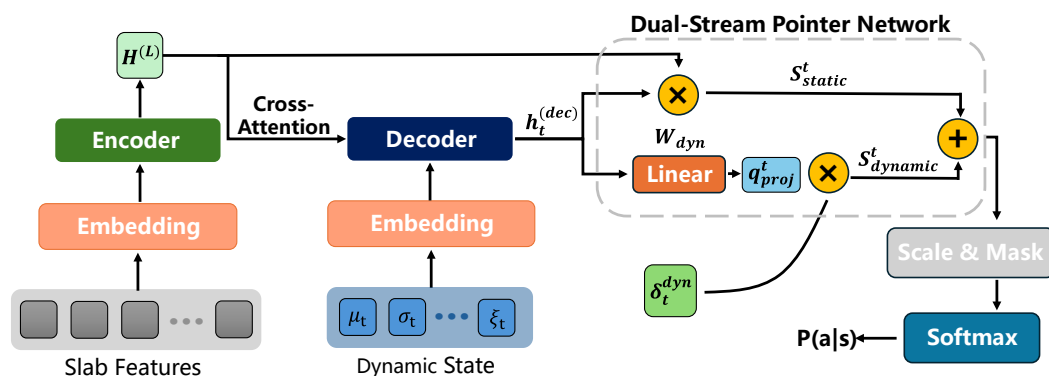


Figure 2. The overall architecture of the DSGO-POMO.

The entire network system operates through the seamless synergy of two core modules. First, a static graph node Encoder performs a one-time extraction of the high-dimensional static features of all candidate slabs. This global representation is fully shared across multiple concurrent rollout trajectories of the same problem instance, thereby drastically reducing the memory overhead of parallel searches. Subsequently, via Cross-Attention, a dynamic Decoder integrated with physical context independently tracks and updates the real-time state and temporal slack for each parallel trajectory. Finally, through a newly constructed Dual-Stream Pointer Network, it autoregressively outputs a strictly legal probability distribution of scheduling actions after passing through the Scale & Mask and Softmax layers.

4.2.1. Static Graph Node Encoder

The core objective of the graph node encoder is to map the discrete, heterogeneous physical attributes of the orders into a high-dimensional continuous latent space representation endowed with a global receptive field. For any node i within the extended candidate set N^+ , its initial input is the 6-dimensional normalized static feature vector x_i as defined in Section 4.1.1.

First, the network elevates the dimensionality of the input features, projecting them into a d_h -dimensional initial node embedding via a learnable linear transformation. Subsequently, this node embedding sequence traverses L stacked standard self-attention encoding blocks. Each block employs a MHA mechanism and a FFN, complemented by standard residual connections and layer normalization to ensure stable gradient propagation. Through the global information interaction facilitated by the multi-layer MHA, the network ensures that any slab can fully perceive the physical attributes and due-date disparities of all other slabs in the candidate pool prior to the commencement of formal scheduling decisions.

After L layers of deep encoding, the network ultimately outputs the high-dimensional node representation matrix H^L embedded with comprehensive global contextual information. During the subsequent decoding operations, this static representation matrix directly serves as the globally shared K and V matrices, providing zero-redundancy foundational feature support for the multi-trajectory parallel inference.

4.2.2. POMO Dynamic Decoder with Heuristic Trajectory Fission

Distinct from the one-time forward inference of the static encoder, the autoregressive decoder must independently track and update unique local physical states for multiple concurrent rollout trajectories within an exceptionally vast solution space. To overcome the rollout catastrophe induced by asymmetric physical constraints in hot rolling scenarios, this article reconstructs the standard decoding process.

Heuristic Trajectory Fission Based on a Prior Token: The standard POMO framework typically relies on uniform traversals across all nodes to initialize parallel trajectories. However, directly migrating this approach to HRPS triggers severe exploration failures. On the one hand, commencing with a narrower slab activates the “non-increasing width” hard constraint, permanently blocking the remaining wider slabs and inducing deadlocks. On the other hand, a prior-free traversal easily overlooks highly time-urgent wide slabs, resulting in inevitable defaults right from the start. These complex dual asymmetric constraints reduce traditional uniform traversal to an inefficient blind search.

To eradicate invalid openings at the source, a heuristic trajectory fission mechanism is introduced at the decoding end. At the initial time step ($t = 1$), the system employs a high-dimensional Learnable Start Token as the initial query vector to output the prior probability distribution of all graph nodes via the cross-attention layer. Updated end-to-end via policy gradients, this token implicitly and adaptively learns the optimal opening trade-off between physical width and due-date urgency. Subsequently, the system employs a Top-K truncation mechanism to precisely screen the top N high-quality nodes with the highest probabilities as valid starting points. These nodes are then independently duplicated M times (i.e., the fission factor) to reconstruct $N \times M$ high-quality concurrent rollout trajectories. Specifically, to prevent homologous trajectories post-fission from collapsing into homogeneity during subsequent action generation, the system introduces a Fission Temperature parameter during the autoregressive sampling phase to dynamically regulate the smoothness of the probability distribution. This mechanism not only ensures the exploration diversity of homologous trajectories within the massive solution space but also perfectly unleashes the model’s potential to circumvent local optima while maintaining the scale of batch optimization.

Dynamic Aggregation of Macroscopic Candidate Pool Features: In the HRPS environment, scheduling a specific slab permanently alters the feasible physical boundaries for subsequent steps due to the stringent “coffin-shaped” (non-increasing width) rule. To prevent the network from making shortsighted local decisions that could trigger irreversible physical deadlocks, the agent must constantly perceive the dynamic dimensional distribution of the remaining inventory. Therefore, the system extracts a lightweight yet

mathematically rigorous statistical sketch—specifically, the physical mean μ_t and maximum absolute deviation σ_t —of the unscheduled candidate pool at each time step t . Let $m_{i,t}$ denote the mask indicating whether slab i remains unvisited, and f_i denote its core physical feature vector (width, thickness, hardness, default penalty). The aggregation formulas are explicitly defined as follows:

$$\mu_t = \frac{\sum_{i=1}^N m_{i,t} f_i}{\sum_{i=1}^N m_{i,t} + \epsilon}, \quad (21)$$

$$\sigma_t = \max_{i \in \{j | m_{j,t}=1\}} |f_i - \mu_t|, \quad (22)$$

where ϵ is an infinitesimally small positive constant. With minimal computational overhead, this feature set constructs a macroscopic statistical representation of the deviation degree of the remaining solution space.

Microscopic Continuous State and Dual Slack Tracking: At the microscopic level, the system tracks the cumulative total length of a single roll u_t , the cumulative identical-width length cl_t , and the cumulative processing time τ_t in real-time. To quantify due-date urgency, the absolute temporal slack for an unscheduled slab j is defined as $\delta_{j,t} = d_j - \tau_t - p_j$. The slacks $\delta_{j,t}$ of all unscheduled slabs jointly constitute the individual dynamic slack vector δ_t^{dyn} , which directly participates in the subsequent pointer scoring mechanism. Meanwhile, to mathematically smooth the severe step-penalties associated with due-date defaults defined in the objective function (Equation 1), the global minimum slack $\delta_t^{min} = \min_j \{\delta_{j,t}\}$ undergoes a truncated negative exponential transformation into an urgency scalar ξ_t :

$$\xi_t = \exp(-\lambda \cdot \max(0, \delta_t^{min})), \quad (23)$$

By employing this specific non-linear transformation, ξ_t remains dormant when temporal slack is abundant, but exponentially amplifies the global risk signal as the tightest time window approaches zero. This mathematical presumption perfectly mirrors the non-linear risk escalation of actual industrial defaults.

Dynamic Context Vector Construction and Cross-Decoding: The system concatenates the embedding of the previously scheduled node $h_{\pi_{t-1}}^L$ (which is the Start Token when $t = 0$), the macroscopic aggregated features, and the microscopic continuous states to construct a fused context query vector c_t at time step t . Subsequently, c_t is linearly projected to serve as the Query and executes a standard cross-attention computation with the global static representation H^L . The resulting decoder hidden state h_t^{dec} is then forwarded into the Dual-Stream Pointer Network to execute the final action scoring.

4.2.3. Dual-Stream Pointer Scoring and Due-Date Intervention

After acquiring the fused context's decoder hidden state h_t^{dec} , standard pointer networks typically rely solely on the inner product between this vector and the global static key matrix to determine action probabilities. However, HRPS inherently harbors a deep-seated adversarial conflict between “smooth physical transitions” and “absolute due-date urgency.” Forcing the model to implicitly perform mental calculations of temporal differences within a high-dimensional latent space easily causes due-date features to be overwhelmed by physical spatial features. Consequently, the model falls into a local optimum, prioritizing a perfect “coffin-shaped” profile while ignoring the tardiness of high-priority orders.

To completely decouple the feature entanglement between physical optimization and temporal optimization, this article profoundly reconstructs the single-stream attention scoring layer and innovatively proposes a Dual-Stream Pointer Scoring mechanism. During the logit calculation phase, this mechanism explicitly introduces a precise mathematical intervention based on individual dynamic slacks.

Static Physical Matching Stream: The first scoring stream aims purely to evaluate the compatibility between the current roll state and the candidate slabs concerning inherent physical attributes such as width, thickness, and hardness. The system executes a dot-product operation between the decoder hidden state and the global key matrix outputted by the static encoder: Based on pattern matching of high-dimensional non-linear features, this score matrix provides the system with a foundational “physical legality” score aimed at minimizing transition penalties.

$$S_{static}^t = h_t^{dec} K^T, \quad (24)$$

Dynamic Temporal Intervention Stream: To circumvent the model’s flaw of blindly estimating elapsed time within the entangled latent space, the second scoring stream establishes a theoretical bridge for explicit spatio-temporal decoupling. It directly incorporates the individual dynamic temporal slack vector δ_t^{dyn} (defined in Section 4.1.1) for precise microscopic targeting. Initially, to formally isolate the temporal intervention from the spatial matching stream, the system mathematically compresses the high-dimensional spatial decoder state (h_t^{dec}) into a single adaptive scalar coefficient via a dedicated projection matrix W_{dyn} and a bias b_{dyn} , followed by a tensor element-wise product with δ_t^{dyn} :

$$q_{proj}^t = h_t^{dec} W_{dyn}^T + b_{dyn}, \quad (25)$$

$$S_{dynamic}^t = q_{proj}^t \odot \delta_t^{dyn}, \quad (26)$$

The essence of this intervention stream is an adaptive temporal weight amplifier driven by end-to-end gradients. The projected scalar q_{proj}^t implicitly encodes the system’s current “temporal urgency coefficient.” Through the penalty feedback of backpropagation, the network spontaneously converges to a negative-weight intervention intelligence: when the macroscopic context warns of severe time scarcity, q_{proj}^t outputs a minimal negative value. By directly multiplying this with the slack, the scores of slabs approaching the default red line ($\delta_{j,t}$) are geometrically amplified, thereby fundamentally endowing the model with precise temporal perception intuition at the architectural level.

Dual-Stream Fusion and Physical Masking: The system linearly superimposes the two streams of scores, thereby achieving a mathematical trade-off between physical smoothness and default avoidance at the tensor level. To prevent the Softmax activation function from entering the saturation zone—which induces gradient vanishing due to excessively large absolute values of temporal slacks—the system applies a constant C for extreme value clipping after executing temperature scaling (using d_{model}):

$$S_{clipped}^t = C \cdot \tanh\left(\frac{S_{static}^t + S_{dynamic}^t}{\sqrt{d_{model}}}\right), \quad (27)$$

Finally, to ensure that the network output strictly complies with the rigorous process restrictions of HRPS, the system directly superimposes the illegal action mask set $\mathcal{M}_t$ (constructed in Section 4.1.3) onto the clipped scores. The corresponding positions of the masked nodes are assigned negative infinity, absolutely zeroing out their transition probabilities. The decoder ultimately outputs the strictly legal scheduling action probability distribution for step t :

$$P(a_t|s_t) = \text{Softmax}(S_{clipped}^t + \mathcal{M}_t), \quad (28)$$

During the training phase, the agent explores the solution space via Monte Carlo sampling based on this probability distribution; during the deployment inference phase, it utilizes active search algorithms to generate the globally optimal scheduling sequence.

4.3. Training and Inference Algorithms

After establishing the underlying Markov Decision Process integrated with physical constraints and the Dual-Stream policy network architecture, the ultimate challenge for industrial deployment lies in guiding the network toward stable convergence within a non-stationary reward space fraught with default penalties, and breaking the inherent solving upper bound of end-to-end models during the deployment phase. To address this, this article introduces GRPO during the training phase and integrates EAS during the inference phase, thereby constructing a closed-loop algorithmic paradigm that boasts both ultra-fast responsiveness and extreme solving optimality.

4.3.1. Group Relative Policy Optimization Training

The underlying foundation of the standard POMO framework relies on the REINFORCE algorithm, utilizing the average return of multiple concurrent trajectories within the same batch as the advantage baseline to estimate policy gradients. However, in the HRPS scenario, the stringent physical state transition barriers and prohibitive due-date default penalties constitute an extremely non-stationary reward space. In such an environment, the REINFORCE algorithm, relying solely on a mean baseline, struggles to accurately delineate the fine-grained relative advantages within the population. Confronted with a massive penalty cliff, the policy gradient—in a bid to circumvent extreme negative feedback—frequently drives the model to adopt overly conservative scheduling actions. This inevitably causes the network to rapidly lose its exploration capability in the early stages of training, thereby precipitating premature convergence and trapping the model deep within a sub-optimal local solution space from which escape is difficult.

To overcome this flaw, this article introduces the GRPO mechanism [34]. It is worth emphasizing that GRPO shares an underlying logical homology with the POMO framework: both rely on the internal statistical features of concurrent rollout trajectories to construct the baseline, thereby entirely discarding the massive and hard-to-converge Critic Network. The system first screens M high-quality candidate starting points via the prior token, and subsequently fissions each starting point K times independently. Consequently, the N ($N = M \times K$) synergistically generated concurrent rollout trajectories directly constitute the sampling population utilized for relative advantage evaluation within the GRPO algorithm.

Nucleus Sampling and Baseline Protection Mechanism: During the autoregressive generation of the trajectory population, to prevent extremely low-probability inferior actions from causing an abnormally high cost for a single trajectory—which would subsequently drag down the cost mean of the entire population and impart erroneous gradient encouragement for “sub-inferior” solutions—the system synergistically incorporates a Top-P nucleus sampling truncation mechanism. Let A_t denote the complete set of legal actions at time step t , sorted in descending order based on their generation probabilities $\pi_\theta(a|s_t)$. The system identifies the minimal high-potential subset A_t^p that satisfies the cumulative probability threshold p : Subsequently, the system performs probability re-normalization and sampling strictly within this subset:

$$\sum_{a \in A_t^p} \pi_\theta(a|s_t) \geq p, \quad (29)$$

$$\pi_\theta(a|s_t) = \begin{cases} \frac{\pi_\theta(a|s_t)}{\sum_{a' \in A_t^p} \pi_\theta(a'|s_t)}, & \text{if } a \in A_t^p, \\ 0, & \text{otherwise} \end{cases} \quad (30)$$

This mechanism effectively prunes the inferior long tail of the solution space, ensuring that the generated trajectory population consistently maintains a high baseline quality. **Population Advantage Evaluation and Trust Region Update:** For the generated trajectory

population $\mathcal{T}$, the system computes the actual total production cost $C(\tau_i)$ for each trajectory τ_i . By calculating the internal cost mean μ_c and standard deviation σ_c of the population, each trajectory is standardized to extract a smooth and precise relative population advantage A_i : where ϵ is an infinitesimally small constant, and c_{max} is a truncation threshold utilized to mask the impact of extreme gradient anomalies.

$$A_i = \text{Clip}\left(\frac{\mu_c - C(\tau_i)}{\sigma_c + \epsilon}, -c_{max}, c_{max}\right), \quad (31)$$

After acquiring the advantages, to prevent destructive updates during a single iteration, this article introduces the clipped surrogate mechanism of PPO, combined with a KL divergence penalty term to restrict the deviation between the new and old policies. Defining the probability ratio of the new to old policy as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, the population maximization surrogate objective function is defined as follows:

$$J_{GRPO}(\theta) = \mathbb{E}_{\tau_i \sim \pi_{\theta_{old}}} \left[\sum_{t=1}^T \min(r_t(\theta)A_i, \text{clip}(r_t(\theta), 1 - \epsilon_c, 1 + \epsilon_c)A_i) \right] - \beta_{KL} D_{KL}, \quad (32)$$

where ϵ_c is the clipping hyperparameter, β_{KL} is the penalty coefficient, and $D_{KL} \approx \frac{1}{2} (\log \pi_\theta - \log \pi_{\theta_{old}})^2$ is the approximate KL divergence.

Dynamic Trust Region Annealing Mechanism: Given that the scheduling agent in this article is trained from scratch (zero-shot), forcibly imposing fixed, minimal trust region boundaries during the initial stages would severely restrict the update step size, causing the model to become trapped in deadlocked solution spaces. Therefore, the system synchronously applies a step-wise decay to the clipping hyperparameter and the tolerated target KL divergence threshold according to the training progression. In the early stages of training, the policy is granted a substantial trial-and-error space to escape traps of invalid policies; as the model gradually converges, the trust region is continuously tightened to provide gradient protection. This mechanism effectively balances early exploration efficiency with late-stage convergence stability.

4.3.2. Inference-Time Optimization via EA-DAS Strategy

To bridge the gap between offline training and zero-shot scale generalization, this article proposes a distribution-aware inference optimization mechanism, termed EA-DAS. Unlike standard EAS, which treats all parameters and instances uniformly, EA-DAS adaptively calibrates the search intensity and parameter plasticity by perceiving the instance-level distribution shift.

Before the search loop, the system first quantifies the “homogeneity” of the current instance by calculating a weighted feature variance V_{batch} . Given the fatal nature of width and due-date constraints, V_{batch} is defined as a weighted sum of their normalized variances: where ω_i represents the domain-prior weight. A lower V_{batch} indicates a highly congested feature space, signaling a high risk of deadlocks and requiring more intensive exploration.

$$V_{batch} = \sum_{i \in \{w, t\}} \omega_i \cdot \sigma_i^2, \quad (33)$$

Asymmetric Differential Updates: To resolve the scale-variant nature of temporal constraints while preserving scale-invariant physical priors, EA-DAS implements an asymmetric parameter update strategy. Specifically:

Fast-Adaptation Group: For out-of-distribution (OOD) scales, the initial Start Token and temporal intervention weights (W_{dyn}) are assigned a higher learning rate to rapidly re-calibrate to the shifted due-date densities.

Fine-Tuning Group: The Normalization layers are restricted to a minimal learning rate to solely absorb latent state drift, while the core physical encoder remains strictly frozen to prevent the “catastrophic forgetting” of rolling rules.

Dual-Annealed Objective Function with Dynamic Floor: The optimization objective is further refined by introducing an entropy-weight annealing mechanism with a dynamic floor. The refined loss function for EA-DAS is formulated as follows, where $\beta(k, V_{batch})$ is the dynamic entropy coefficient at iteration k . Crucially, instead of decaying to zero, β is constrained by a minimum exploration floor $\beta_{min}(V_{batch})$: This minimum floor keeps the search trajectories diverse enough for local adjustments, even during the final iterations. This stops the population from converging into identical copies. Ultimately, these adaptive steps allow EA-DAS to find more accurate schedules and generalize better across different problem sizes than standard end-to-end decoding.

$$J_{EA-DAS}(\phi) = \mathbb{E}_{\tau \sim \pi_{\phi}}[C(\tau)] - \beta(k, V_{batch}) \cdot \mathbb{E}_{\tau \sim \pi_{\phi}}[\mathcal{H}(\pi_{\phi})], \quad (34)$$

$$\beta(k, V_{batch}) = \max(\beta_{min}(V_{batch}), \beta_{init} \cdot (1 - k/K)), \quad (35)$$

5. Experiments

5.1. Experimental Setup and Baseline Algorithms

5.1.1. Experimental Environment and Parameter Settings

To ensure absolute reproducibility and strictly simulate the actual cyber–physical conditions of industrial hot rolling, a comprehensive instance generation mechanism was constructed. The physical attributes of the slabs (e.g., width, thickness, hardness) and the corresponding transition penalty weights were independently sampled from specific distributions derived from real-world metallurgical scenarios. Furthermore, the hard time-window boundaries (due dates) were dynamically generated based on a load-aware equation controlled by an urgency factor θ . Due to space constraints, the detailed mathematical formulations, distribution intervals, and physical boundary thresholds for generating these instances are comprehensively provided in Appendix A (Table A1 and Equation (A1)) of this article.

All computational experiments in this study were executed on a dedicated engineering workstation. The hardware infrastructure comprises an Intel Core i9-13900HX CPU (Intel Corporation, Santa Clara, CA, USA), 16 GB of DDR5 4800MHz RAM (Kingston Technology, Fountain Valley, CA, USA), and an NVIDIA GeForce RTX 4060 Laptop GPU (NVIDIA Corporation, Santa Clara, CA, USA) equipped with 8 GB of VRAM. On the software side, the framework was implemented using Python 3.12.12 and PyTorch 2.5.1, utilizing CUDA 12.1 for GPU acceleration. Additionally, Gurobi Optimizer 13.0.1 was deployed to solve the mathematical program and establish the exact theoretical baselines.

For the proposed DSGO-POMO framework, the neural network structure and training parameters were systematically calibrated across different problem scales (n). To balance the exploration capacity and computational efficiency consistently, the scale-dependent sampling hyperparameters (i.e., Valid Starts and Fission Factor) were proportionally adjusted to ensure adequate and fair training coverage across all scales. To enhance research transparency and facilitate flawless reproducibility, the comprehensive hyperparameter configurations of the environment, model architecture, and training phase are systematically summarized in Table 2.

Table 2. Hyperparameter settings for the DSGO-POMO framework across different problem scales.

Category	Hyperparameter/Component	Value/Setting
Environment & Sampling	Problem Scale (n)	10/20/50/100
	Valid Starts	5/5/10/15
	Fission Factor	5/7/10/20
	Total Parallel Trajectories	25/35/100/300

Model Architecture	Embedding Dimension (d_h)	128
	Feed-Forward Dimension	512
	Encoder Layers/Attention Heads	6 Layers/8 Heads
	Logit Clipping Factor	10
Training Configuration	Training Episodes per Epoch	10,000
	Batch Size	512/512/256/128
	Optimizer/Base Learning Rate	AdamW/ 10^{-4}
	LR Decay Milestones	[85%, 95%] $\times$ Total Epochs

5.1.2. Baseline Algorithms and Comparative Strategy Settings

To comprehensively evaluate the solving quality and generalization performance of the proposed algorithm across instances of varying scales, this article selects three categories of highly representative baseline algorithms for multi-dimensional comparison. To eliminate the stochastic variance inherent in traditional randomized algorithms, all heuristic algorithms are executed independently 10 times on each test instance, and the best objective function value achieved is retained for the final comparison. The specific baseline settings are as follows:

Exact Solver: The commercial optimizer Gurobi is utilized to find the true global optima. Because HRPS is NP-hard, Gurobi scales poorly as the problem size grows. Therefore, it is only executed on very small instances ($n = 10$) to establish a perfect theoretical baseline.

Classical Operations Research Heuristics: To comprehensively evaluate the solving quality and generalization capability of the proposed DSGO-POMO framework, standard implementations of three representative meta-heuristic algorithms were selected as baselines: Hybrid Genetic Algorithm (HGA), Max–Min Ant System (MMAS), and Variable Neighborhood Search (VNS). These methods were strategically chosen as they represent three fundamentally distinct search paradigms—evolutionary computation, swarm intelligence, and local trajectory exploitation, respectively—which are widely adopted in the literature for solving complex scheduling and vehicle routing problems [5,6,11].

Rather than utilizing highly over-fitted and customized variants, the classical, standard versions of these algorithms were independently developed to ensure a fair and reproducible benchmark. Crucially, the standard frameworks of these heuristics were explicitly adapted and strictly bounded by the physical masking rules (e.g., the coffin-shaped profile constraint) to guarantee the generation of 100% physically feasible HRPS sequences.

- **HGA:** Leverages global crossover and mutation operators to maintain population diversity and avoid premature convergence.
- **MMAS:** Utilizes a time-aware heuristic and strictly bounded pheromone trails to effectively construct sequential solutions under asymmetric physical constraints.
- **VNS:** Employs multiple progressively destructive neighborhood structures to escape local optima in heavily coupled spatio-temporal search spaces.

The core mechanism parameters for these comparative methods are summarized in Table 3.

Table 3. Core hyperparameter configurations for the heuristic algorithms.

Algorithm	Hyperparameter/Mechanism	Value/Setting
HGA	Crossover Rate/Mutation Rate	0.8/0.2
MMAS	Pheromone Factor/Heuristic Factor	1.0/2.0
	Evaporation Rate	0.05
VNS	Neighborhood Structures (Max = 4)	Swap, Insert, Inversion, Block-Move

Deep Reinforcement Learning Baseline: The universally recognized milestone architecture in this domain, POMO [18], is selected. To guarantee the rigor of the comparison, the core hyperparameters of POMO, such as the underlying feature dimension and the number of network layers, are kept strictly consistent with the proposed model. As a pure learning-based benchmarking algorithm, POMO is utilized to objectively validate the effectiveness of the proposed Dual-Stream network and GRPO mechanism in navigating the rigorous constraints of hot rolling.

Through comprehensive comparisons with the aforementioned exact solver, traditional heuristic baselines, and the classical DRL baseline, the proposed algorithm exhibits exceptional solving precision and ultra-fast inference performance in the subsequent experiments across all scales.

5.1.3. Computational Complexity Analysis

To systematically respond to the theoretical efficiency of the hot rolling production scheduling problem (HRPS) and the proposed solution framework, the computational complexity is analyzed from two dimensions: the problem's mathematical scale and the algorithm's inference boundary.

- (1) **Complexity of the HRPS Problem:** As previously stated, the HRPS investigated in this study is mathematically modeled as a highly constrained prize-collecting vehicle routing problem (PCVRP). Since the standard VRP is inherently strongly NP-hard, the HRPS, which couples strict asymmetric physical constraints (e.g., “coffin-shaped” width profiles and cumulative roll fatigue), exhibits even greater computational intractability. For an instance with n candidate slabs, the worst-case solution space scales factorially as $\mathcal{O}(n!)$, inevitably leading to a combinatorial explosion. Consequently, exact mathematical solvers (e.g., Gurobi) suffer from an exponential time explosion ($\mathcal{O}(2^n)$), and traditional meta-heuristics inevitably hit severe computational bottlenecks on larger instances. This makes them highly impractical for real-time online rescheduling—such as when equipment fails or a rush order arrives—where feasible, high-quality sequences must be generated within strict and narrow time windows.
- (2) **Complexity of the DSGO-POMO Framework:** Once the offline training phase is completed, the online execution of the proposed DSGO-POMO framework relies entirely on deterministic neural forward network inference. The theoretical computational burden is strictly dominated by the network components described in Section 4:

Encoder Phase: For n slabs, the multi-head self-attention (MHA) mechanism in a single encoder layer requires computing the query-key-value dot-products, yielding a time complexity of $\mathcal{O}(n^2 \cdot d_h)$, where $d_h = 128$ represents the hidden embedding dimension. With $L = 6$ stacked encoder layers, the total encoder complexity is strictly bounded by $\mathcal{O}(L \cdot n^2 \cdot d_h)$.

Decoder Phase: The decoder sequentially constructs the rolling schedule autoregressively via n decision steps. At step t , computing the attention weights between the last scheduled node and the remaining unvisited nodes scales as $\mathcal{O}((n - t) \cdot d_h)$. Cumulatively over n steps, the decoding process requires $\sum_{t=1}^n \mathcal{O}((n - t) \cdot d_h) = \mathcal{O}(n^2 \cdot d_h)$.

In summary, the total online inference time complexity of DSGO-POMO is strictly a polynomial-time bound of $\mathcal{O}(L \cdot n^2 \cdot d_h)$. In the practical implementation, this is further accelerated via Pytorch's FlashAttention contiguous memory kernel optimization. This low-order polynomial complexity theoretically guarantees that the framework can deliver high-quality production schedules within milliseconds. It fundamentally overcomes the computational bottlenecks of traditional metaheuristics, fully satisfying the stringent speed requirements for dynamic, real-time industrial rescheduling.

5.2. Overall Scheduling Performance and Scale Generalization Analysis

5.2.1. Optimality Validation on Small-Scale Instances

In this section, the capability of the proposed algorithm to approximate the theoretical global optimal solutions is evaluated via extremely small-scale instances. The experiment randomly generated 10 hot rolling scheduling instances of size $n = 10$, and the solving objective values from the commercial solver Gurobi and the proposed model were statistically recorded.

The computational results demonstrate that, coupled with the EAS online optimization strategy, the comprehensive scheduling costs outputted by the proposed end-to-end RL model perfectly match the optimal values from Gurobi across all 10 instances, achieving a 100% optimal solution hit rate. By perfectly matching the exact solver on small instances, it is confirmed that the model obeys all hard constraints and successfully finds the global optimum.

5.2.2. Performance on In-Distribution Instances

The algorithms are evaluated on in-distribution instances of size $n \in \{20, 50, 100\}$. For $n = 20$ and $n = 50$, 1000 independently generated instances are tested to ensure statistical reliability. However, since traditional heuristics execute too slowly on larger problems, the test set is limited to 10 instances for $n = 100$. To ensure a fair comparison, maximum search times for the three heuristics (HGA, MMAS, and VNS) are strictly set based on the problem size: 15 s for $n = 20$, 60 s for $n = 50$, and 600 s for $n = 100$. These limits provide the baseline algorithms with ample time to fully explore the search space. The detailed empirical results for these experiments are summarized in Table 4.

Table 4. Comprehensive Performance Comparison on In-Distribution Instances.

Method	$n = 20$		$n = 50$		$n = 100$	
	Cost	Time	Cost	Time	Cost	Time
HGA	9969.6	11.52 m	23,859.5	46.18 m	42,853.5	10.01 m
MMAS	9637.1	11.53 m	23,218.6	46.30 m	43,383.7	10.01 m
VNS	9586.2	11.51 m	23,646.4	46.02 m	43,523.6	10.00 m
POMO	10,237.3	0.01 m	22,086.6	0.01 m	40,918.3	0.03 m
DSGO-POMO (Greedy)	9792.2	0.03 m	20,818.1	0.09 m	38,034.7	0.03 m
DSGO-POMO (EA-DAS)	9586.1	0.66 m	20,609.9	7.82 m	37,512.1	0.76 m

The empirical results clearly demonstrate that the proposed DSGO-POMO framework comprehensively outperforms all baseline methods in both solution quality and computational efficiency across all problem scales. On small-scale instances ($n = 20$), while the standard POMO model (cost: 10,237.3) underperforms compared to traditional heuristics, the incorporation of the EA-DAS strategy enables the proposed framework (cost: 9586.1) to establish a highly competitive bound, marginally surpassing the best-performing heuristic, VNS (cost: 9586.2).

More importantly, as the combinatorial complexity scales to $n = 50$ and $n = 100$, the superiority of the proposed model becomes overwhelmingly pronounced. Even relying solely on pure greedy decoding, DSGO-POMO consistently achieves lower-cost solutions than all heuristic baselines—for instance, realizing a 10.3% cost reduction compared to MMAS at $n = 50$. The application of EA-DAS further refines these solutions, establishing a dominant performance gap.

In terms of computational efficiency, the advantage of the DRL paradigm is transformative. At $n = 50$, while traditional heuristics require approximately 46 min to

converge, the greedy DSGO-POMO model completes the inference in a mere 0.09 min. Even with the active search fine-tuning of EA-DAS (7.82 min), the total computational latency remains fundamentally shorter than that of the heuristics, perfectly satisfying the strict real-time response requirements of dynamic shop-floor scheduling.

Furthermore, to rigorously verify that the algorithmic superiority of the proposed framework is not influenced by stochastic variance, a Wilcoxon Signed-Rank Test was conducted across 1000 testing instances. The statistical analysis compared the final comprehensive costs of the proposed DSGO-POMO framework (with EA-DAS) against the baseline algorithms. While the proposed method demonstrates highly competitive bounds on small-scale scenarios ($n = 20$), its statistical dominance becomes overwhelming as the combinatorial complexity scales up to $n = 50$. Specifically, the p -values resulting from the Wilcoxon tests against HGA, MMAS, and VNS at the $n = 50$ scale are 5.28×10^{-165} , 9.11×10^{-165} , and 2.01×10^{-165} , respectively. Since these p -values converge to near-zero and are strictly less than the standard significance level of 0.05, it is definitively confirmed that the performance improvements and computational stability achieved by the proposed dual-stream architecture are highly statistically significant.

As theoretically established in Section 4.2.3, this overwhelming empirical superiority fundamentally stems from the proposed dual-stream architecture. Unlike traditional heuristics that waste computational resources on physically infeasible local traps, and standard DRL models that suffer from spatio-temporal feature entanglement, the explicitly decoupled framework ensures highly stable gradient flows, perfectly explaining why DSGO-POMO consistently achieves near-optimal convergence with exceptional efficiency.

5.2.3. Zero-Shot Generalization to Large-Scale Instances

In practical industrial hot rolling production, the scale of the order pool frequently fluctuates due to dynamic demands. Constrained by hardware GPU memory limitations during model training, individually training specific weights for large-scale tasks across all possible scales lacks engineering feasibility. Therefore, this section evaluates the model's zero-shot generalization capability when confronted with significant distribution shifts. By directly freezing the model weights trained and converged at the $n = 100$ scale, cross-scale testing is conducted on 10 newly generated ultra-large-scale instances of $n = 200$. To guarantee that traditional heuristic algorithms possess ample capability to explore the solution space, their maximum search time is uniformly set to 30 min.

In the zero-shot generalization tests conducted on ultra-large-scale instances of $n = 200$, the proposed model exhibits remarkable cross-scale extrapolation capabilities and computational efficiency advantages. As illustrated in Table 5, despite directly freezing the $n = 100$ training weights without any scale-specific fine-tuning, the proposed model achieves an average scheduling cost of 74,314.1 under the pure greedy decoding (Greedy) mode. This demonstrates superior solving quality compared to traditional heuristic algorithms that were granted 30 min of ample iterative time (e.g., HGA at 85,980.5, MMAS at 88,216.5, and VNS at 91,144.3). Furthermore, its computational time of 0.04 min (approximately 2.4 s) achieves a lead of several orders of magnitude.

Table 5. Large-Scale Generalization Performance on $n = 200$ Instances.

Method	$n = 200$	
	Cost	Time
HGA	85,980.5	30.05 m
MMAS	88,216.5	30.01 m
VNS	91,144.3	30.03 m
POMO	80,135.2	0.04 m

DSGO-POMO (Greedy)	74,314.1	0.04 m
DSGO-POMO (EA-DAS)	73,238.7	1.72 m

With the further incorporation of the EA-DAS online optimization mechanism, the model performs lightweight fine-tuning—averaging approximately 10.32 s per instance (totaling 1.72 min)—to further suppress the average comprehensive cost down to 73,238.7. This yields a performance improvement of approximately 14.7% over HGA, the best-performing heuristic baseline.

These results show that the Dual-Stream network and GRPO training effectively learn the core physical rules and scheduling patterns of hot rolling. Because the model captures these fundamental dynamics rather than memorizing a specific size, its performance remains stable across different problem scales. In contrast to the search efficiency bottlenecks encountered by traditional heuristic algorithms when facing ultra-large-scale solution spaces, the end-to-end model coupled with inference-time search strategies can more robustly breach the solving upper bounds within complex, constrained spaces while ensuring blazing-fast real-time responsiveness.

5.2.4. Convergence and Training Dynamics Analysis

To comprehensively evaluate the authentic contributions of each component within the proposed architecture to the optimization dynamics and stability, this section selects two representative instance scales, $n = 50$ and $n = 100$, to compare the training trajectories of the standard POMO model, the ablation model incorporating only GRPO, and the proposed full model (Dual-Stream + GRPO) over the first 200 epochs. Based on the morphological characteristics depicted in Figure 3a,b, a detailed analysis is presented below.

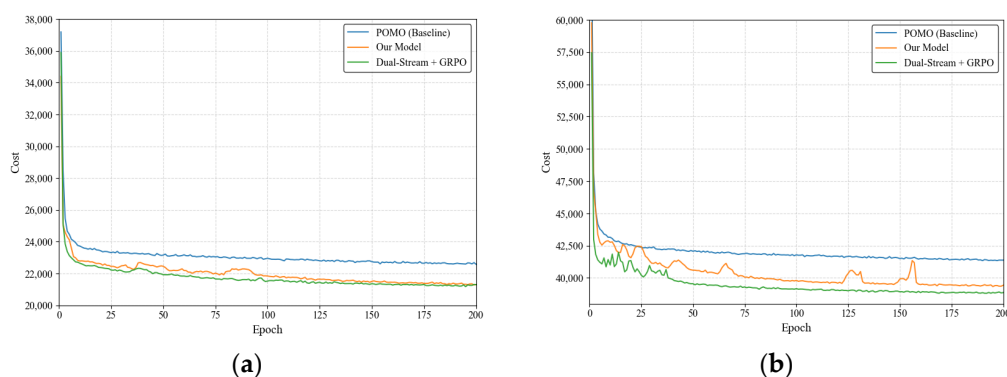


Figure 3. Convergence curves of different models during the training phase. (a) Convergence performance on medium-scale instances; (b) Convergence performance on large-scale instances.

In the first 25 epochs, the training curves drop at very different rates. Standard POMO descends slowly. In contrast, both the GRPO-only model and the full model show sharp cost reductions. This early behavior highlights a key flaw in the traditional REINFORCE baseline: it cannot provide effective gradient guidance under the high-variance penalties of hot rolling. By calculating relative advantages within the population, GRPO gives the model much clearer direction right from the start, speeding up early convergence.

As training continues, the models behave differently. The standard POMO curve flattens out too early, getting stuck at higher costs (around 23,000 for $n = 50$ and 41,500 for $n = 100$). Because its single-stream baseline lacks feature decoupling, POMO plays it safe to avoid extreme penalties, trapping itself in sub-optimal local minima.

The GRPO-only model explores more aggressively but suffers from heavy optimization oscillations. For instance, on the 100-slab task, it hits severe cost spikes around epochs 125 and 155. Without explicit feature decoupling, the policy clearly struggles to balance physical hard constraints and due-date penalties. It tries to escape local optima but fails to navigate the vast infeasible regions smoothly.

Finally, the full model maintains the most stable and optimal trajectory. Adding the Dual-Stream network successfully tames the late-stage spikes seen in the pure GRPO model, helping the curve settle earlier and reach the lowest final cost across both problem scales. By physically decoupling the spatial constraints from the temporal features at the architectural level, the Dual-Stream design eliminates feature entanglement. Together with GRPO's efficient exploration, the complete framework avoids premature convergence and reliably finds high-quality solutions, breaking through the optimization bottlenecks of traditional policy gradients.

5.3. Sensitivity Analysis and Domain-Specific Search Dynamics

To fully address the structural superiority of the proposed DSGO-POMO architecture and evaluate the time-quality trade-off, rigorous ablation experiments were conducted on the core hyper-parameters.

5.3.1. Resource Allocation and Domain-Specific Search Dynamics

As introduced in the GRPO mechanism, the proposed framework relies on screening high-quality candidate starting points (valid_starts, N_s) via the prior token, and subsequently fissioning each point (fission_factor, F_c) independently to construct the concurrent rollout population. To explicitly determine the optimal balance between this starting breadth and fission depth, comprehensive ablation studies were conducted under the pure Greedy mode.

The results in Table 6 structurally validate the parameter selection from two distinct perspectives:

1. **Robustness under Fixed Budgets (≈ 100 concurrent trajectories):** Attempting a uniform allocation strategy ($N_s = 50, F_c = 2$) wastes computational power on inferior starting points, yielding a poor cost of 20,882.53. Conversely, while the extreme elite configuration ($N_s = 5, F_c = 20$) achieves a marginally lower cost (20,709.18), it heavily over-exploits the initial screening of the prior token. In highly constrained HRPS scenarios, if the true optimal sequence stems from a lower-ranked starting node due to unforeseeable physical barriers, this extreme strategy prematurely prunes viable branches. This risks trapping the model deep within a sub-optimal local solution space from which escape is difficult. Thus, the (10,10) configuration acts as a robust Pareto optimal point. It gladly trades a negligible cost difference for double the starting state diversity, effectively preventing over-fitting to the prior token's initial bias.
2. **Marginal Returns of Trajectory Fission (Fixed $N_s = 10$):** By isolating the fission factor, the Δ column visually captures a classic diminishing marginal return trend. Compared to the baseline without fission ($F_c = 1$), introducing a moderate fission factor ($F_c = 10$) profoundly reduces the cost by 387.28 units. This confirms that independently fissioning the starting points synergistically empowers the model to conquer the downstream combinatorial complexity. However, further pushing F_c to 20 triggers diminishing returns—reducing the cost by merely 54.41 units while noticeably inflating the execution time to 4.80 s.

Table 6. Sensitivity Analysis of Start Selection and Trajectory Fission Configurations.

Valid Starts	Fission Factor	Total Trajectories	Average Cost	Time (s)	Cost Reduction (Δ)
50	2	100	20,882.53	4.55	-
25	4	100	20,836.89	3.79	-
5	20	100	20,709.18	3.69	-
10	1	10	21,129.42	2.39	Baseline
10	5	50	20,835.13	2.64	-294.29
10	10	100	20,742.14	3.63	-92.99
10	20	200	20,687.73	4.80	-54.41

Consequently, (10,10) is formally established as the default hyper-parameter configuration, ensuring maximum worst-case robustness and physical constraint satisfaction while maintaining millisecond-level responsiveness.

5.3.2. Sensitivity Analysis on EA-DAS Iterations

Building upon the robust 10, 10 configuration, the time-quality trade-off of the inference-time optimization mechanism, the EA-DAS strategy, was further evaluated on large-scale instances ($n = 50$). As presented in Table 7, the sensitivity analysis of the active search epochs visually captures the algorithm's convergence dynamics and marginal returns ($\Delta Cost / \Delta Time$).

Table 7. Sensitivity Analysis of EA-DAS Epochs under Default Configuration ($N_s = 10$, $F_c = 10$).

EA-DAS Epochs	Average Cost	Total Time (s)	Cost Reduction (Δ)	Time (s)
0 (Greedy)	21,094.68	1.06	20,882.53	4.55
10	20,830.44	89.88	-264.24	2.97
30	20,722.72	297.92	-107.72	0.52
50 (Optimal)	20,609.91	469.38	-112.81	0.66
80	20,510.95	734.90	-98.96	0.37

Note: The bold values indicate the optimal configuration identified through our experimental analysis.

The trajectory of the marginal returns reveals a compelling narrative about the solution space. In the initial phase (0 to 10 epochs), the EA-DAS mechanism rapidly calibrates the temporal intervention parameters, causing the schedule cost to drop precipitously by 264.24 units. This yields an exceptionally high marginal return of 2.97 cost units reduced per second, proving that the model can swiftly escape initial local optima traps with minimal search steps.

Interestingly, the marginal return exhibits a secondary surge at 50 epochs (rising from 0.52 to 0.66). This counter-intuitive peak suggests that the EA-DAS strategy, driven by its entropy-weight annealing mechanism, successfully breached a deep local optimum barrier during this phase, unlocking a highly optimized region of the combinatorial space. Immediately after this breakthrough (at 80 epochs), the marginal return collapses to a mere 0.37. Extending the search consumes an additional 265 s but yields severely diminishing returns, indicating that the algorithm has fully converged and the search space is largely exhausted.

Therefore, stopping at exactly 50 epochs captures this critical secondary breakthrough while strictly avoiding the inefficient computational overhead of the long-tail convergence. It is formally selected as the optimal Pareto balance for offline heavy-search tasks, whereas the pure Greedy mode (0 epochs, 1.06 s) remains the ultimate blazing-fast paradigm for real-time dynamic rescheduling.

6. Conclusions

In summary, this paper has investigated the Hot Rolling Production Scheduling (HRPS) problem under strict physical and due-date constraints. To overcome the computational latency of traditional meta-heuristics and the training instability of standard DRL models, a Mixed-Integer Programming model is formulated, and a novel end-to-end scheduling framework named DSGO-POMO is proposed. At its core, the Dual-Stream network explicitly decouples physical routing rules from temporal features, successfully preventing the generation of deadlocked sequences. During offline training, a Group Relative Policy Optimization (GRPO) mechanism is integrated to stabilize policy updates within the high-penalty environment. For online inference, an EA-DAS strategy is applied to further refine the output schedules.

Extensive computational experiments confirm the exceptional performance of DSGO-POMO. On small instances, it perfectly matches the global optima found by exact solvers. On medium-scale tasks, it significantly outperforms classical meta-heuristics in both solution quality and execution time. Furthermore, when extrapolated to ultra-large out-of-distribution instances, the pre-trained model demonstrates robust zero-shot generalization, establishing a blazing-fast rescheduling paradigm.

Despite these promising results, this study acknowledges certain limitations. First, while the problem instances are generated based on strict physical boundaries derived from real-world scenarios, the framework has not yet been deployed using raw, real-time production logs from an actual steel enterprise due to strict commercial data confidentiality. Second, the current model focuses exclusively on the single-stage scheduling optimization of the hot rolling phase, whereas actual metallurgical production is a highly coupled multi-stage process.

Consequently, future research will primarily focus on two directions. First, efforts will be made to establish deeper industrial collaborations to evaluate and fine-tune the proposed framework against real-time dynamic disruptions (e.g., equipment breakdowns) on authentic shop floors. Second, the single-stage model will be expanded into a multi-stage collaborative framework, potentially leveraging multi-agent reinforcement learning to synchronously optimize the entire steel production pipeline from continuous casting to hot rolling.

Author Contributions: Conceptualization, C.W. and W.C.; methodology, C.W. and W.C.; software, C.W.; validation, C.W. and W.C.; formal analysis, C.W.; investigation, C.W.; resources, W.C.; data curation, C.W.; writing—original draft preparation, C.W.; writing—review and editing, C.W., W.C. and M.H.; visualization, C.W.; supervision, W.C. and M.H.; project administration, W.C. and M.H.; funding acquisition, M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the NSFC [grant number 92567302], and the Fundamental Research Funds for the Central Universities [grant number N25ZLV008].

Data Availability Statement: The data presented in this study are available on request from the corresponding author. Details regarding the data generation process and instance scales are provided in the Appendix of this article.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature: A comprehensive list of mathematical symbols, variables, and parameters utilized throughout the problem formulation is summarized below:

Sets & Indices	Description
N	The set of physical candidate slab nodes pending scheduling, defined as $N = \{1, 2, \dots, n\}$.
0	The dummy start and end node, representing the idle state of equipment or a roll-changing operation.
N^+	The extended scheduling network set including the dummy node, formulated as $N^+ = N \cup \{0\}$.
i, j	Node indices within the extended scheduling network, where $i, j \in N^+$.
Parameters	
Z	The objective function value representing the total comprehensive production cost to be minimized.
w_i	The core physical width attribute of slab $i \in N^+$, where the dummy node width is set to $w_0 = +\infty$.
g_i	The core physical thickness attribute of candidate slab $i \in N$.
h_i	The core physical hardness attribute of candidate slab $i \in N$.
l_i	The physical rolling length of candidate slab $i \in N$.
p_i	The processing time of slab $i \in N$, which is numerically equivalent to l_i under constant speed.
e_i	The earliest allowable start time (release date) for processing candidate slab $i \in N$.
d_i	The latest due date (hard time-window completion boundary) for candidate slab $i \in N$.
pv_i	The default penalty cost incurred if candidate slab $i \in N$ is excluded from the final scheduling plan.
α	The penalty weight coefficient assigned to abrupt jumps in width between adjacent processed slabs.
β	The penalty weight coefficient assigned to abrupt jumps in thickness between adjacent processed slabs.
γ	The penalty weight coefficient assigned to abrupt jumps in hardness between adjacent processed slabs.
$c_{i,j}$	The single-step transition penalty cost caused by physical attribute jumps when slab j follows slab i .
C_0	The inherent fixed capital cost required to initiate a new rolling unit (i.e., roll-changing cost).
T_{setup}	The fixed time expenditure required to execute a physical roll-changing operation.
L_{max}	The maximum fatigue physical length limit allowed within the lifecycle of a single rolling unit.
L_{min}	The minimum economic operational length threshold required prior to a roll-changing operation.
L_w	The maximum allowable cumulative length for continuous local wear of identical slab widths.
M	An arbitrarily large positive real number utilized as the penalty factor in Big-M inequalities.
$I(\cdot)$	An indicator function that outputs 1 if the inner condition is satisfied, and 0 otherwise.
Variables	
$x_{i,j}$	Binary decision variable: 1 if slab j is processed immediately after slab i ; 0 otherwise.
y_i	Binary decision variable: 1 if candidate slab i is selected into the scheduling plan; 0 otherwise.
u_i	Continuous state variable recording the cumulative processing length of the current roll up to node i .
cl_i	Continuous state variable recording the cumulative length of consecutively processed slabs with identical widths up to node i .
τ_i	Continuous state variable recording the absolute system time when the processing of slab i commences.

Appendix A

Appendix A.1. Parameter Distribution of Instances

The HRPS instances utilized in the experiments of this article are generated based on simulated data distributions from industrial hot rolling production. The physical attributes and economic default penalty parameters for each candidate slab i were independently sampled from specific uniform distributions $U[\min, \max]$. Additionally, the transition penalty weights for physical attributes are set as empirical constants. The detailed parameter settings are presented in Table A1.

Table A1. Parameter distributions of the generated HRPS instances.

Parameter	Symbol	Distribution/Value
Width	w_i	$U[1.0, 5.0]$

Thickness (Gauge)	g_i	$U[1.0,5.0]$
Hardness	h_i	$U[1.0,5.0]$
Rolling Length	l_i	$U[20.0,40.0]$
Default Penalty	pv_i	$U[1200.0,1800.0]$
Transition Weights	α, β, γ	Constant: 20.0

Appendix A.2. Physical Constraint Boundaries

To strictly adhere to the practical production safety standards of the metallurgical industry, the model establishes rigorous hard constraint boundaries. The relevant physical thresholds are defined as follows:

Fatigue limit of a single roll ($L_{\max}$): The cumulative processing length of a single rolling unit must not exceed 200. Minimum economic operational length ($L_{\min}$): To guarantee production profitability, the cumulative length prior to executing a roll-changing operation must reach at least 50. Localized wear limit (L_w): The maximum cumulative length for consecutively rolling slabs of identical width is strictly restricted to 100. Roll-changing time overhead (T_{setup}): The fixed downtime required to execute a roll-changing operation is set to 10.

Appendix A.3. Due-Date Generation Mechanism

This article adopts a dynamic time-window generation method based on the total load of the order pool. Let $\theta = 0.3$ denote the urgency factor. The latest due date d_i for slab i is uniformly and randomly sampled within the following interval:

$$d_i \sim U\left(10.0 + p_i, 10.0 + p_i + 2(1 - \theta) \sum_{j=1}^n p_j\right), \quad (\text{A1})$$

where p_i represents the processing time of slab i (which is numerically equivalent to its length l_i), and $\sum_{j=1}^n p_j$ denotes the total processing time of all slabs under the current problem scale n .

Reference

1. Lopez, L.; Carter, M.W.; Gendreau, M. The hot strip mill production scheduling problem: A tabu search approach. *Eur. J. Oper. Res.* **1998**, *106*, 317–335.
2. Cowling, P.I.; Ouelhadj, D.; Petrovic, S. A multi-agent architecture for dynamic scheduling of steel hot rolling. *J. Intell. Manuf.* **2003**, *14*, 457–470.
3. Mao, K.; Pan, Q.-K.; Chai, T.; Luh, P.B. An effective subgradient method for scheduling a steelmaking-continuous casting process. *IEEE Trans. Autom. Sci. Eng.* **2014**, *12*, 1140–1152.
4. Tan, Y.; Zhou, M.; Wang, Y.; Guo, X.; Qi, L. A hybrid MIP–CP approach to multistage scheduling problem in continuous casting and hot-rolling processes. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1860–1869.
5. Tang, L.; Wang, X. Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. *Int. J. Adv. Manuf. Technol.* **2006**, *29*, 1246–1258.
6. Chen, Y.-W.; Lu, Y.-Z.; Ge, M.; Yang, G.-K.; Pan, C.-C. Development of hybrid evolutionary algorithms for production scheduling of hot strip mill. *Comput. Oper. Res.* **2012**, *39*, 339–349.
7. Jia, S.; Yi, J.; Yang, G.; Du, B.; Zhu, J. A multi-objective optimisation algorithm for the hot rolling batch scheduling problem. *Int. J. Prod. Res.* **2013**, *51*, 667–681.
8. Tang, L.; Wang, X. An improved particle swarm optimization algorithm for the hybrid flowshop scheduling to minimize total weighted completion time in process industry. *IEEE Trans. Control. Syst. Technol.* **2009**, *18*, 1303–1314.
9. Jiang, S.; Liu, M.; Hao, J. A two-phase soft optimization method for the uncertain scheduling problem in the steelmaking industry. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *47*, 416–431.
10. Tang, L.; Wang, X. A scatter search algorithm for a multistage production scheduling problem with blocking and semi-continuous batching machine. *IEEE Trans. Control. Syst. Technol.* **2010**, *19*, 976–989.

11. Zhang, R.; Song, S.; Wu, C. Robust scheduling of hot rolling production by local search enhanced ant colony optimization algorithm. *IEEE Trans. Ind. Inform.* **2019**, *16*, 2809–2819.
12. Zhang, R.; Song, S.; Wu, C. The hot strip mill scheduling problem with uncertainty: Robust optimization models and solution approaches. *IEEE Trans. Cybern.* **2022**, *53*, 4079–4093.
13. Jiang, S.-L.; Liu, Q.; Cao, L.-L.; Sun, L. An online learning-based mACO approach for hot rolling scheduling problems involving dynamic order arrivals. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 12322–12335.
14. Jiang, S.-L.; Liu, Q.; Bogle, I.D.L.; Zheng, Z. A self-learning based dynamic multi-objective evolutionary algorithm for resilient scheduling problems in steelmaking plants. *IEEE Trans. Autom. Sci. Eng.* **2022**, *20*, 832–845.
15. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. 10.48550/arXiv.1506.03134
16. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv* **2016**, arXiv:1611.09940.
17. Kool, W.; Van Hoof, H.; Welling, M. Attention, learn to solve routing problems! *arXiv* **2018**, arXiv:1803.08475.
18. Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; Min, S. Pomo: Policy optimization with multiple optima for reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21188–21198.
19. Song, W.; Chen, X.; Li, Q.; Cao, Z. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Trans. Ind. Inform.* **2022**, *19*, 1600–1610.
20. Lei, K.; Guo, P.; Zhao, W.; Wang, Y.; Qian, L.; Meng, X.; Tang, L. A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem. *Expert Syst. Appl.* **2022**, *205*, 117796.
21. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208.
22. Pan, R.; Wang, Q.; Cao, J.; Zhou, C. Deep reinforcement learning for solving steelmaking-continuous casting scheduling problems under time-of-use tariffs. *Int. J. Prod. Res.* **2024**, *62*, 404–420.
23. Kong, M.; Wang, W.; Deveci, M.; Zhang, Y.; Wu, X.; Coffman, D.M. A novel carbon reduction engineering method-based deep Q-learning algorithm for energy-efficient scheduling on a single batch-processing machine in semiconductor manufacturing. *Int. J. Prod. Res.* **2024**, *62*, 6449–6472.
24. Park, J.; Chun, J.; Kim, S.H.; Kim, Y.; Park, J. Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning. *Int. J. Prod. Res.* **2021**, *59*, 3360–3377.
25. Zhang, C.; Song, W.; Cao, Z.; Zhang, J.; Tan, P.S.; Chi, X. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1621–1632.
26. Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; Song, L. Learning combinatorial optimization algorithms over graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3488–3498.
27. Xu, Y.; Fang, M.; Chen, L.; Xu, G.; Du, Y.; Zhang, C. Reinforcement learning with multiple relational attention for solving vehicle routing problems. *IEEE Trans. Cybern.* **2021**, *52*, 11107–11120.
28. Zhu, K.; Li, X.; Jiang, P.; Cheng, M.; Wu, Y.; Gao, K.; Ren, L. Distributed Flexible Job Shop Scheduling With Heterogeneous Transportation Resources Constraints via Deep Reinforcement Learning and Graph Neural Network. *IEEE Trans. Syst. Man Cybern. Syst.* **2026**, *56*, 3086–3098.
29. Che, G.; Zhao, S.; Zhao, G.; Zhang, Y. Reinforcement learning for energy-efficient multi-objective dynamic planning of hot rolling production. *J. Manuf. Syst.* **2026**, *86*, 356–375.
30. Su, X.; Li, M.; Li, Q.; Chen, C.; Guo, S.; Wang, X. Energy consumption prediction for manufacturing in industrial IoT based on heterogeneous GNN. In Proceedings of the 2024 IEEE/CIC International Conference on Communications in China (ICCC), Hangzhou, China, 7–9 August 2024; pp. 627–632.
31. Akashwar, G.; Balaji, M. IoT-Based Data Acquisition and ML Forecasting for Intelligent Control of Solar Energy Microgrids. In Proceedings of the 2026 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSSES), Chennai, India, 29–30 January 2026; pp. 1–8.
32. Zhao, L.; Fan, J.; Zhang, C.; Shen, W.; Zhuang, J. A DRL-based reactive scheduling policy for flexible job shops with random job arrivals. *IEEE Trans. Autom. Sci. Eng.* **2023**, *21*, 2912–2923.

33. Hottung, A.; Kwon, Y.-D.; Tierney, K. Efficient active search for combinatorial optimization problems. *arXiv* **2021**, arXiv:2106.05126.
34. Guo, D.; Yang, D.; Zhang, H.; Song, J.; Wang, P.; Zhu, Q.; Xu, R.; Zhang, R.; Ma, S.; Bi, X. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* **2025**, *645*, 633–638.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.