Computers and Chemical Engineering 204 (2026) 109381

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/cace

Bilevel scheduling in downstream oil supply chain: Integrating reinforcement learning with mathematical programming☆

Qipeng Yang [a,b,c], Wentian Fan [a,b,c], Nan Ma [f,g], Shu Lin [d,e,*], Jiawen Chang [f,g], Zhiqiang Zou [a,c], Liang Sun [f,g], Haifeng Zhang [b,c,d,e]

[a] Nanjing University of Posts and Telecommunications, No. 9 Wenyuan Road, Yadong New District, Nanjing, 210023, Jiangsu, China
[b] Nanjing Artificial Intelligence Research of IA, No. 266 Chuangyan Road, Jiangning District, 211135, Nanjing, Jiangsu, China
[c] University of Chinese Academy of Sciences, Nanjing, No. 188 Tianquan Road, Jiangning District, 211135, Nanjing, Jiangsu, China
[d] National Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institution of Automation, Chinese Academy of Sciences, No. 95 Zhongguancun East Road, Haidian District, Beijing, 100190, China
[e] Institute of Automation Chinese Academy of Sciences, No. 95 Zhongguancun East Road, Haidian District, Beijing, 100190, China
[f] Petrochina Planning & Engineering Institute, No. 3 Zhixin West Road, Haidian District, Beijing, 100083, China
[g] Key Laboratory of Oil & Gas Business Chain Optimization, CNPC, No. 3 Zhixin West Road, Haidian District, Beijing, 100083, China

ARTICLE INFO

Keywords:
Oil supply chain scheduling
Reinforcement learning
Mathematical programming
Rolling-Horizon method
Simulation-base evaluation

ABSTRACT

With the growth of global energy demand, optimizing the oil supply chain has become crucial. This paper proposes a hybrid reinforcement learning (RL) and mathematical programming (MP) scheduling approach to optimize downstream oil supply chain operations, including refinery production scheduling, logistics distribution, and inventory management. This approach decomposes the complex problem into multiple sub-problems using a Rolling-Horizon method (RH), enhancing computational efficiency and flexibility. We conduct a comparative analysis to evaluate two RL training algorithms with RH: Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) denoted as SAC-RH and PPO-RH respectively. Experimental results from the simulation-based evaluation demonstrate that the SAC version excels in handling complex dynamic environments and continuous action space problems, significantly reducing the number of early warnings and improving overall optimization results. This study demonstrates the applicability of RL in industrial automation and identifies potential avenues for future research.

1. Introduction

Global economic growth and increasing energy demand have made oil a key resource increasingly important in modern industry. According to the IEA Oil Market Report (International Energy Agency (IEA), 2024), global oil supply is expected to reach a new record of 104.5 million barrels per day by 2025. An efficient oil supply chain is crucial to ensuring stable production, market supply and operating costs.

The oil supply chain encompasses a complex set of links from upstream to downstream, such as exploration, production, crude oil transportation, refining, petrochemical processing, consumer goods manufacturing, storage and logistics management (Hussain et al., 2006). The upstream supply chain involves the acquisition of crude oil, including exploration, forecasting, production, and logistics management of crude oil transport from remote wells to refineries. The downstream part of the supply chain — including production scheduling, logistics distribution, and inventory management of the refinery — is directly related to the supply of refined oil products, cost control, and market responsiveness. To maximize economic efficiency, refineries need to consider market demand, price fluctuations, and other factors to develop production scheduling plans (Saharidis et al., 2009). By optimizing demand forecasting, distribution channels, transportation routes, warehouse management, and information flow, operating costs can be effectively reduced and profitability and flexibility can be improved throughout the supply chain (Lisitsa et al., 2019).

Crude oil scheduling activities (e.g., procurement, fractionation schemes, feed distribution, and production scheduling) typically involve long time cycles and are upper-level and long-process production and business activities. These activities encompass complex industrial

☆ This article is part of a Special issue entitled: 'Reinforcement Learning' published in Computers and Chemical Engineering.

* Corresponding author.
E-mail addresses: yangqipeng23@mails.ucas.ac.cn (Q. Yang), fanwentian23@mails.ucas.ac.cn (W. Fan), manan2013@petrochina.com.cn (N. Ma), shu.lin@ia.ac.cn (S. Lin), changjiawen_wl@petrochina.com.cn (J. Chang), zouzq@njupt.edu.cn (Z. Zou), sunliang_cppei@petrochina.com.cn (L. Sun), haifeng.zhang@ia.ac.cn (H. Zhang).

https://doi.org/10.1016/j.compchemeng.2025.109381
Received 14 April 2025; Received in revised form 25 June 2025; Accepted 29 August 2025
Available online 8 September 2025

constraints, increasing the scale and complexity of the problem (Kelly and Mann, 2003). As the time span grows, so does the amount of data processed, including historical data, predictive information, and observation data. Multiple decision points, such as offloading and mixing, are usually studied separately, and it is difficult to reach the global optimal solution (Li et al., 2020).

In a real-world business scenario, the oil and gas industry chain usually has multiple links, complex structure, limited adjustment space, and the production process often shows strong "rigidity". This characteristic makes the production operation face the impact of internal and external uncertainties (Al-Othman et al., 2008), and MP methods are difficult to meet the requirements of fast solutions for complex large-scale problems.

Given the above challenges, MP methods often face challenges of low computational efficiency and poor model scalability when solving large-scale multivariable optimization problems. However, contemporary artificial intelligence technologies, particularly machine learning and RL, are changing this status quo. It has become a trend to improve the scheduling efficiency and optimization level of the crude oil industry chain through data-driven methods, enable rapid generation of scheduling solutions, and provide decision support for emergency response (Makkar et al., 2020).

In order to address these challenges, we develop a bilevel optimization approach for downstream oil supply chain, coupling RL with MP. This approach aims to provide rapid solution generation for the downstream oil supply chain scheduling problem, especially to adjust dynamic strategies to uncertainties and emergencies. Combining traditional MP methods with modern artificial intelligence technologies, the approach helps to achieve more reasonable resource allocation and more efficient emergency response, and comprehensively improve the operation level of downstream links in the oil industry chain. Key contributions of this paper include:

- Integrating reinforcement learning with mathematical programming in scheduling: the scheduling approach combines RL with MP to improve the efficiency of solving large and complex scheduling optimization problems. This approach can not only handle high-dimensional sparse linear programming problems, but also dynamically adapt to market changes and emergencies to achieve efficient scheduling adjustments.
- Balancing urgent demands and planning stability via bilevel Rolling-Horizon method: The RH method is a widely used strategy in control theory and optimization, known for its ability to handle complex dynamic systems by performing repeated optimizations at each sampling time, enabling real-time adjustments. In this study, the RH method is applied to downstream scheduling to decompose long-term planning into a sequence of manageable short-term sub-problems. At the high level, RL determines the target inventory for each node, while at the low level, a mathematical solver performs detailed operational optimization based on these targets within the RH framework. This integrated approach ensures both rapid responsiveness to urgent demands and the consistency and stability of long-term planning.
- Simulation-based algorithm evaluation in real-world business scenarios: a simulation environment based on real data provides inventory to the MP layer every day, accepts scheduling plans from the RL layer to simulate real business scenarios. This simulator provides a solid foundation for verifying the effectiveness and robustness of scheduling algorithms.

The remainder of the paper is structured as follows: Section 2 summaries the existing methods for oil supply chain scheduling. Section 3 formally describes the downstream oil supply chain scheduling problem and defines the corresponding mathematical model. Section 4 presents the innovative scheduling method which integrates RL with MP, and applies Bilevel RH algorithm for optimization. Section 5 demonstrates the evaluation results by simulating in real-world business scenarios. Section 6 concludes the paper and proposes several improvement suggestions for future work.

2. Related work

The existing scheduling methods for downstream oil supply chain include mathematical programming (Section 2.1), heuristic search (Section 2.2), and reinforcement learning (Section 2.3).

2.1. Mathematical programming methods

In the early days, refinery operations relied mostly on manual planning methods and spreadsheet models for scheduling, which was a time-consuming process and did not use optimization tools to aid decision-making (Göthe-Lundgren et al., 2002). With the development of automation technologies, the technology based on MP has brought new methods to solve the problem of supply chain scheduling optimization.

Al-Othman et al. (2008) introduced a multi-period stochastic programming model to enhance the pressure resistance of production planning by considering market demand and price uncertainty. However, accurately modeling market uncertainty and dealing with computational burdens remains a challenge, especially in volatile markets. The robust optimization framework proposed by Li et al. (2012) combined with continuous time modeling significantly improves the ability to cope with demand fluctuations, but the actual deployment complexity is high and requires a large amount of computing resources.

The nonlinear programming and mixed integer optimization models developed by Joly et al. (2002) increase the yield of high-value products while meeting product specification constraints. Such models face computational complexity and resource consumption in real-time or large-scale applications. The Mixed Integer Nonlinear Programming model proposed by Moro and Pinto (2004) is suitable for short-term crude oil scheduling and effectively deals with inventory management problems, but it is difficult to solve in large-scale instances and consumes high computational resources. The work from Mouret et al. (2011) decomposes the complex Mixed integer nonlinear programming problem (MINLP) into more manageable subproblems through Lagrangian decomposition, improving the robustness of the algorithm and the quality of the solution. Nonetheless, due to the large number of binary variables and non-convex constraints involved, MINLP solvers may struggle to converge or find only suboptimal solutions. Li et al. (2012) proposed a continuous-time MINLP model based on unit-specific events to optimize crude oil operation scheduling. This method simplifies the non-convex problem by using the piecewise linear underestimation technique, but still requires a lot of computational resources to solve the complex non-convex MINLP problem.

2.2. Heuristic search methods

When solving large or complex optimization problems, it takes a very long time for accurate algorithms based on MP to find the optimal solution, which is often impractical in practical applications. In contrast, the heuristic search methods can provide a near-optimal solution in a shorter period of time, which greatly improves the solution efficiency (Sahebi et al., 2014). Hou et al. (2015) proposed to use genetic algorithms to optimize detailed scheduling schemes, simplifying the complex scheduling problem to the allocation problem of feeding tanks and retorts. Although this method is effective, its performance is highly dependent on the selection of the initial population and the setting of parameters, which limits the possibility of finding the global optimal solution to a certain extent.

On the other hand, Adhitya et al. (2007) developed a new heuristic rescheduling strategy that generates new viable scheduling schemes with minimal configuration changes by decomposing any scheduling into operational chunks, and preserving those chunks as much as possible. Although this method may sacrifice the degree of optimization to a certain extent, that is, the final scheduling scheme may not be absolutely optimal, it significantly improves the solution speed and

flexibility. In addition, the study of Abdolazimi and Abraham (2021) explores a multi-objective oil supply chain model based on meta-heuristics under both uncertain and deterministic conditions. In order to solve the challenges of minimizing transportation costs and loads, they adopted particle swarm optimization and multi-objective particle swarm optimization under deterministic and uncertain conditions, as well as the Mulvey method. The results show that the proposed model can effectively solve the problem in both cases.

However, when employing heuristics for decision-making, it is important to note that the effectiveness of the model depends largely on the quality of the input data. Inaccurate or outdated data can lead to sub-optimal decisions that affect the final solution quality (Bányai et al., 2015).

2.3. Reinforcement learning methods

With the advancement of artificial intelligence, particularly RL, the field of supply chain optimization has witnessed transformative progress (Dai et al., 2021; Rolf et al., 2023). RL learns to maximize cumulative rewards through environmental interaction, demonstrating unique advantages in high-uncertainty scenarios requiring rapid adaptation, such as short-term crude oil scheduling and dynamic decision-making. This data-driven paradigm complements traditional mathematical programming methods, which, despite providing structured optimal solutions for oil supply chain planning, often suffer from static assumptions and computational intractability in highly dynamic environments.

Mathematical programming and heuristic approaches dominate oil supply chain planning, but Kim et al. (2024) highlight their limitations through an AI-driven two-layer framework: The upper layer employs Markov decision processes to capture long-term trends (e.g., declining equipment investment costs, evolving carbon tax policies); The lower layer uses linear programming for hourly scheduling, balancing wind power fluctuations and load demands. This framework bridges data-driven and optimization algorithms, enabling a shift from “static planning” to “dynamic adaptation” with significant advantages in long-term uncertainty management and system-wide collaborative optimization.

In the petroleum sector, reinforcement learning (RL) has emerged as a cornerstone for uncertainty-aware decision-making, firmly grounded in the actor-critic (AC) paradigm. This paradigm serves as the theoretical bedrock for modern algorithms such as Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018). The inherent environmental stochasticity, however, poses a core challenge. Specifically, the state transition probabilities $P(s_{t+1|s_t,a_t})$ and reward distributions $R(s_t, a_t)$ are subject to fluctuations, much like the unpredictable variations in renewable energy generation and load demands. To address this, RL leverages its online learning mechanisms, which continuously update decision-making strategies based on real-time feedback.

The effectiveness of RL in handling such challenges is closely tied to the evolution of the AC framework. The AC framework originated from the need to balance policy gradient and value-based methods. Early implementations like the classic Actor-Critic with eligibility traces (Konda and Tsitsiklis, 1999) introduced dual components:

- An actor (policy network) for action selection.
- A critic (value network) for action evaluation via value function estimation.

This architecture addressed the high variance of pure policy gradients and the continuous-action limitations of value-based methods. Subsequent advancements — such as Advantage Actor-Critic and Asynchronous Advantage Actor-Critic (Mnih et al., 2016) — improved sample efficiency, while Deep Deterministic Policy Gradient extended AC to continuous action spaces.

Recent studies further advance RL applications in petroleum scenarios:Dell’Aversana (2024) propose an RL framework integrating self-awareness and reflection mechanisms, enabling agents to evaluate performance and dynamically adjust network architectures for optimized oilfield production management. Zahedi-Seresht et al. (2024) utilize Q-learning to determine optimal initial production and injection rates, maximizing oil recovery efficiency. Zavvari and Saifoddin Asl design a two-stage framework: deep learning for oil price prediction in the first stage, followed by RL combined with multi-criteria decision-making for optimized allocation from crude oil to refined products.

3. Problem description and modeling

In this section, we first describe the constraint model for optimizing production operations in downstream oil supply chain. As shown in Fig. 1, the downstream oil supply chain mainly includes refinery nodes, transfer nodes, provincial library nodes, and sale nodes, which together constitute the key links in the downstream scheduling and optimization process of the oil supply chain, and determine the operational efficiency of oil product processing, storage and transportation tasks. In order to gain an in-depth understanding of the downstream production operations in the oil supply chain, the role of each type of nodes in the supply chain and their corresponding constraints are described in detail from Sections 3.1 to 3.3. Section 3.4 presents the constraints on transport edges while Section 3.5 defines the objective function of the constraint model. In Table 1, we summarize the key variables and parameters used in the proposed optimization model.

3.1. Constraints on refinery nodes

The refinery nodes are the primary processing link downstream in the oil supply chain, and are primarily responsible for the conversion of crude oil into refined products (e.g. gasoline, diesel), as well as inventory management and transportation of refined products. The operation of the refinery nodes is highly continuous and complex, and its production schedule directly affects the flow and coordination of products throughout the downstream supply chain.

The actual operating capacity of the refinery nodes is limited by the actual production capacity, which is the amount of crude oil that can be processed per unit of time. The processing capacity of a refinery node can be dynamic, and it will be affected by factors such as fluctuations in raw material supply and market demand. Therefore, in our scheduling model, the production planning of the refinery nodes needs to take these factors into account to ensure that the processing task meets the sales demand while avoiding the constraints of its production capacity. Inventory management at the refinery nodes includes crude oil inventory management and refined product inventory management. Crude oil inventories determine whether refineries have continuous production capacity, while refined product inventories affect the supply capacity of downstream transfer nodes and sale nodes. The storage capacity of all oil products in the refinery node should be kept within a reasonable range to avoid additional costs due to over-storage or downstream supply chain disruptions caused by insufficient inventory. Our model will take these factors into account to help refinery nodes make more accurate decisions to ensure a balance between production and sales, in order to maximize economic benefits.

We define the set of refinery nodes as V_r. For each refinery node $v \in V_r$, crude oil can be processed into different grades of gasoline and diesel fuel. Assuming that the daily crude oil processing capacity for refinery node v is dp_v^t, and the processing rates for converting crude oil into gasoline and diesel are rg_v and rd_v respectively, the production process yields a quantity of gasoline with grade i denoted as pg_{vi}, and a quantity of diesel with grade i denoted as pd_{vi}.

In refinery node v, the initial inventory of crude oil is dc_v^0 and the other time steps are represented as dc_v^t. The inventory of gasoline

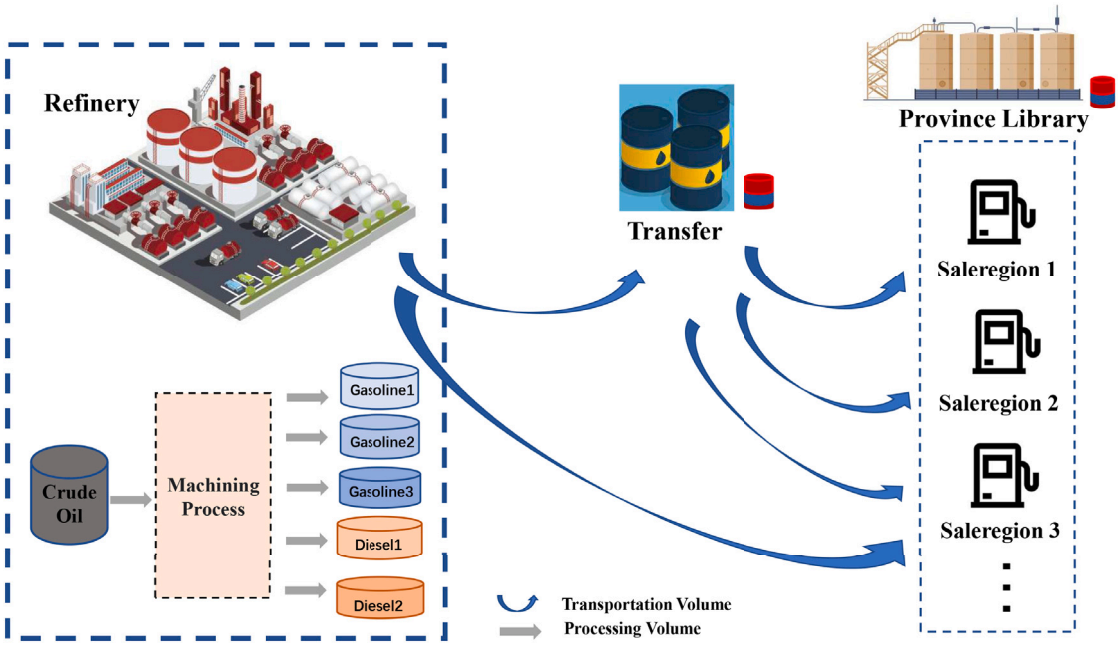

Fig. 1. An overview of the downstream oil supply chain.

Table 1
Parameters and variables summary.

Category	Symbol	Node/Edge type	Meaning
Refinery nodes (V_r)			
Parameters	V_r	–	Refinery node set
	dp_v^t	Refinery	Daily crude processing capacity
	rg_v, rd_v	Refinery	Crude→ gasoline/diesel conversion rates
	$Rlc_v, Rslc_v$	Refinery	Crude inventory (lower, safe lower bound)
	$Rsuc_v, Ruc_v$	Refinery	Crude inventory (safe upper, upper bound)
	$Rslg_v, Rsug_v$	Refinery	Gasoline inventory (safe lower, safe upper bound)
	Rug_v, Rud_v	Refinery	Gasoline/Diesel inventory upper bound
	$Rsld_v, Rsud_v$	Refinery	Diesel inventory (safe lower, safe upper bound)
	g_v^t	Refinery	Daily crude supply (Day 1: Real; Days 2+: RL)
Variables	pg_{vi}^t, pd_{vi}^t	Refinery	Grade-i gasoline/diesel production (Day t)
	dc_v^t	Refinery	Crude inventory (Day t)
	dgv_i^t, ddv_i^t	Refinery	Grade-i gasoline/diesel inventory (Day t)
Transfer nodes (V_t)			
Parameters	V_t	–	Transfer node set
	Tug_v, Tud_v	Transfer	Gasoline/diesel storage upper bounds
	$Tslg_v, Tsug_v$	Transfer	Gasoline inventory (safe lower, safe upper bound)
	$Tsld_v, Tsud_v$	Transfer	Diesel inventory (safe lower, safe upper bound)
Variables	dg_v^t, dd_v^t	Transfer	Total gasoline/diesel inventory (Day t)
Provincial library nodes (V_p)			
Parameters	V_p	–	Provincial library node set
	ng_{vi}^t, nd_{vi}^t	Provincial	Daily grade-i gasoline/diesel demand (Day t)
	$Pslg_v, Psug_v$	Provincial	Gasoline inventory (safe lower, safe upper bound)
	Pug_v, Pud_v	Provincial	Gasoline/Diesel inventory upper bound
	$Psld_v, Psud_v$	Provincial	Diesel inventory (safe lower, safe upper bound)
Variables	dg_{vi}^t, dd_{vi}^t	Provincial	Grade-i gasoline/diesel inventory (Day t)
Transport edges (E)			
Parameters	E, E_g, E_d	–	Transport edge sets (total, gasoline, diesel)
	$p_{e,v}^t$	Edge	Daily transport capacity upper bound (Edge e, Node v)
	c_e	Edge	Transport cost (Edge e)
Variables	$f_e^t, f_e^{t-t_e}$	Edge	Transport volume (Day t, daily/cross-day)

and diesel with grade i are denoted as dg_{vi}^t, dd_{vi}^t. Use u to denote the upper bound, su the upper safety bound, l the lower bound, and sl the lower safety bound. The inventories of these three types of oils must all remain within safe inventory limits. and for diesel inventory, $dd_{vi}^t \in [0, Rud_v]$. The volume of transportation from a refinery node to a transfer node or a sale node is denoted as f_e^t. Therefore, for refinery node v each day ($t = 1, 2, 3, \ldots$), the processing volume and inventory constraints are formally defined from Eqs. (1) to (7).

$$\frac{pg_v^t}{rg_v} + \frac{pd_v^t}{rd_v} \le dp_v^t \tag{1}$$

$$dc_v^t = dc_v^{t-1} + g_v^t - \left(\frac{pg_v^t}{rg_v} + \frac{pd_v^t}{rd_v}\right) \tag{2}$$

$$\sum_i^{N_g} dg_{vi}^t = \sum_i^{N_g} dg_{vi}^{t-1} + pg_{vi}^t - \sum_{e=\langle v,v'\rangle \in E_g} f_e^t \tag{3}$$

$$\sum_i^{N_g} dd_{vi}^t = \sum_i^{N_g} dd_{vi}^{t-1} + pd_{vi}^t - \sum_{e=\langle v,v'\rangle \in E_d} f_e^t \tag{4}$$

$$dc_v^t \in [max(Rlc_v, Rslc_v), min(Rsuc_v, Ruc_v)] \tag{5}$$

$$max(0, Rslg_v) \le \sum_i^{N_g} dg_{vi}^t \le min(Rsug_v, Rug_v) \tag{6}$$

$$max(0, Rsld_v) \le \sum_i^{N_g} dd_{vi}^t \le min(Rsud_v, Rud_v) \tag{7}$$

In Eq. (2), g_v^t represents the daily supply of crude oil at refinery node v. The value g_v^1 on the first day is taken from real data, and the values $g_v^2, g_v^3, \ldots$ on subsequent days are suggested by RL.

3.2. Constraints on transfer nodes

Transfer nodes are important bridges connecting refinery nodes and provincial library nodes. They typically represent large oil depots or intermediate stations, which have capability for storing various types of refined products, ensuring effective management of these products. In the constraint model, the storage capacity of transfer nodes is also limited by the depot capacity. We define the set of transfer nodes as V_t. For each transfer node $v \in V_t$, it can store gasoline and diesel transported from refinery nodes, and the refined products from refinery nodes can be transported to the provincial library nodes, with the transport volume denoted as f_e^t ($f_e^{t-t_e}$ indicates cross-day transportation). The storage capacity of gasoline and diesel in the transfer nodes must be maintained within a safe storage range. Assuming the storage capacities for gasoline and diesel in transfer nodes are Tug_v and Tud_v respectively, then for each transfer node v, the safety storage constraints for each day ($t = 1, 2, 3, \ldots$) are given from Eqs. (8) to (11).

$$dg_v^t = dg_v^{t-1} + \sum_{e=\langle v',v\rangle \in E_g} f_e^{t-t_e} - \sum_{e=\langle v,v'\rangle \in E_g} f_e^{t-t_e} \tag{8}$$

$$dd_v^t = dd_v^{t-1} + \sum_{e=\langle v',v\rangle \in E_d} f_e^{t-t_e} - \sum_{e=\langle v,v'\rangle \in E_d} f_e^{t-t_e} \tag{9}$$

$$max(0, Tslg_v) \le dg_v^t \le min(Tsug_v, Tug_v) \tag{10}$$

$$max(0, Tsld_v) \le dd_v^t \le min(Tsud_v, Tud_v) \tag{11}$$

3.3. Constraints on provincial library nodes

Provincial library nodes are distribution centers within an area (e.g., a province or a state), consisting of multiple sale nodes, primarily responsible for the terminal supply and demand satisfaction of refined products. Provincial library nodes collaborate with refinery nodes and transfer nodes to manage the storage of refined products by setting reasonable inventory capacities for different types of oil, ensuring a balance in refined product storage and supply across various regions. If a sale node experiences a shortage, the corresponding provincial library node can quickly dispatch refined products to maintain regional supply stability. The intelligent scheduling model needs to have early warning function to monitor inventory levels in real-time and automatically generate emergency scheduling plans when inventory approaches critical thresholds, ensuring that the demands of sale nodes are met properly. We define the set of provincial library nodes as V_p. For each provincial library node $v \in V_p$, it can store gasoline and diesel of different grades. Assume the daily demand for grade i gasoline at provincial library node v is ng_{vi}^t and the daily demand for brand i diesel is nd_{vi}^t. Then, for each node $v \in V_p$, the safety inventory constraints for each day ($t = 1, 2, 3, \ldots$) are given by the following functions:

$$\sum_i^{N_g} dg_{vi}^t = \sum_i^{N_g} dg_{vi}^{t-1} + \sum_{e=\langle v',v\rangle \in E_g} f_e^{t-t_e} - ng_v^t \tag{12}$$

$$\sum_i^{N_g} dd_{vi}^t = \sum_i^{N_g} dd_{vi}^{t-1} + \sum_{e=\langle v',v\rangle \in E_d} f_e^{t-t_e} - nd_v^t \tag{13}$$

$$max(0, Psug_v) \le \sum_i^{N_g} dg_{vi}^t \le min(Psug_v, Pug_v) \tag{14}$$

$$max(0, Psud_v) \le \sum_i^{N_g} dd_{vi}^t \le min(Psud_v, Pud_v) \tag{15}$$

3.4. Constraints on transport edges

There are multiple transport edges between refinery nodes, transfer nodes, and provincial library nodes. Each edge corresponds to an actual transportation path between different nodes. For each edge $e \in E$, gasoline and diesel of brand i can be transported. Assume that the transportation capacity of each path e is limited to p_e^t (10,000 tons per day), and the transportation cost is c_e (CNY per 10,000 tons). Then, for any single transportation volume, it must satisfy the following constraint:

$$\sum_{e=\langle v,v'\rangle \in (E_g \cup E_d)} f_e^t \in [0, p_{e,v}^t] \tag{16}$$

In Eq. (16), the edge $e = (v, v')$ represents a directed link for transporting either gasoline ($e \in E_g$) or diesel ($e \in E_d$), denoted by $e \in E_g \cup E_d$.

3.5. Objective function

The objective function is designed to minimize the total transportation costs of all oil products and to balance the difference between actual inventory and target inventory. The function is as follows:

$$\min \sum_t f_e^t c_e + \beta \sum_t g^t \tag{17}$$

where $\sum_t f_e^t c_e$ represents the transportation cost, $\sum_t g^t$ represents the penalty function for the difference between actual inventory and target inventory, and β is a balancing coefficient used to balance the importance of the two indicators.

4. Methodology

In the previous section, we define the constraint optimization model of the downstream oil supply chain scheduling problem. In this section, we will introduce our approach for solving the optimization problem, which consists of the MP solving technology, the RL algorithm, and the bilevel RH method.

4.1. Mathematical programming solving

Commercial solvers are specialized software tools designed to address a wide range of optimization problems, from simple Linear Programming to complex Mixed-Integer Nonlinear Programming. For instance, Gurobi (Gurobi Optimization, 2024) and Cplex (2009) are two highly efficient and widely adopted commercial solvers. Gurobi is renowned for its speed in handling large-scale linear and mixed-integer linear problems, while CPLEX, developed by IBM, supports a broad spectrum of optimization types, including linear, mixed-integer, and quadratic programming. These capabilities make CPLEX suitable for enterprise-level applications such as supply chain management and financial portfolio optimization.

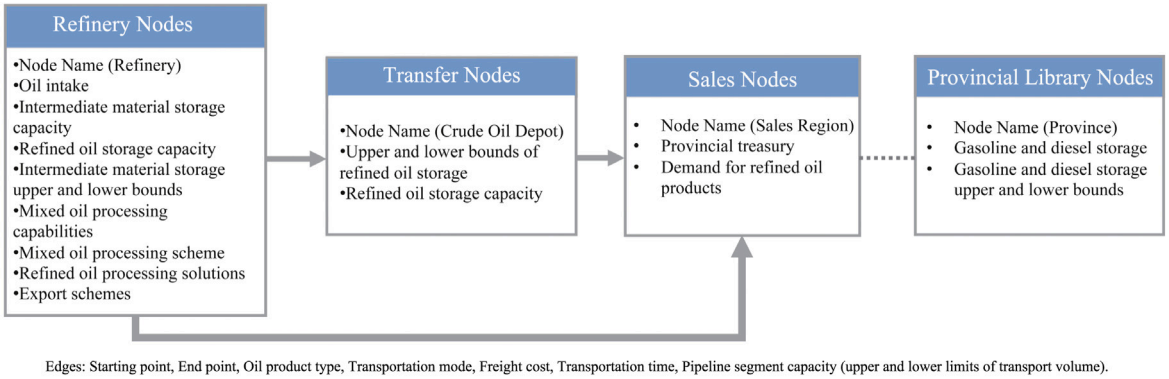

Fig. 2. List of node information extracted from raw data.

In this study, we construct a constraint optimization model for downstream oil supply chain scheduling based on 17 Excel sheets of raw data provided by China National oil Corporation. Through a series of customized data loading and parsing methods, we extracted information from these tables covering all aspects of the supply chain, including downstream supply node details, inventory status, refinery operational details, and the capacity of the transportation network. The formulated mathematical model is then implemented and solved using Gurobi to obtain optimal scheduling strategies that satisfy all operational constraints.

In the data processing phase, we design specialized parsing methods for different types of data. As shown in Fig. 2, for the information of the downstream supply nodes, we extract the production cycle, the type of oil and its production limit; For the refinery nodes, we analyze the oil intake plan, processing plan and the upper and lower limits of the processing volume; In terms of transport networks, key factors such as modes of transport, cost and timeliness between different routes are identified. All processed data are integrated into a structured configuration dictionary, which provides a solid basis for subsequent supply chain optimization.

Variables, constraints, and the objective function of the constraint optimization model are defined using the detailed data described above to simulate the supply chain network. Specifically:

- Variables. A variety of decision-making variables are set according to actual demand, such as the daily crude oil input of each refinery node, refined oil production, intermediate material processing volume, etc. Factors such as inventory levels and transportation flows are also considered.
- Constraints. Multiple constraints are set in order to reflect real-world constraints, including refinery capacity, inventory capacity caps, mass balance requirements, and flow limitations along the transport path.
- Objective function. An objective function is set to minimize the total cost or maximize the profit, and in particular, by piecewise linearization of storage expenses, the system is encouraged to find a solution that satisfies all constraints while minimizing costs.

4.2. Reinforcement learning

RL is a method where an agent learns to achieve goals by interacting with an environment (Kaelbling et al., 1996). This interaction is iterative, with the agent's ultimate goal being to maximize the expected cumulative reward obtained through multiple episodes of interaction. The specific manner of interaction between the agent and the environment is illustrated in Fig. 3. In each episode, the agent perceives the current state s_t of the environment at a given timestep, then computes and executes an action a_t. This action is applied to the environment, which subsequently generates the corresponding reward

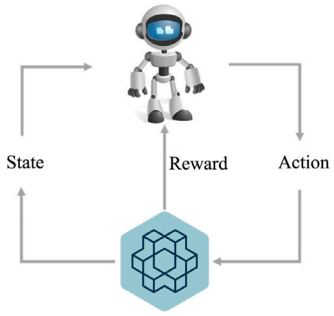

Fig. 3. The basic framework of reinforcement learning.

signal r_t and transitions to a new state. The agent then observes the updated environmental state s_{t+1} at the next timestep, and the process continues. Through repeated interactions across episodes, the agent learns an optimal policy that enables it to make decisions that maximize long-term cumulative rewards.

We can describe the process of RL as a Markov Decision Process. An MDP is formally defined as a tuple $M = (S, A, P, r, \gamma)$, where:

- S is the state space,
- A is the action space,
- $P(s'|s,a) : S \times S \times A \to [0,1]$ is the transition probability function, representing the probability of transitioning from state s to state s' after applying action a,
- $r : S \times A \to \mathcal{R}$ is the reward function,
- $\gamma \in (0,1)$ is the discount factor.

In the context of RL, strategies include deterministic policies and stochastic policies. Suppose a policy π is a probability distribution defined on $S \times A \to \mathcal{R}$, where $\pi(a|s)$ denotes the probability of selecting action a at state s. Specifically, it is described as:

$$\pi(a|s) = P(A_t = a|S_t = s) \tag{18}$$

Let Π denote the set encompassing all possible policies. In an MDP, due to the presence of the Markov property, the strategy needs to depend only on the current state and does not need to consider historical states.

We express the state-value function of the reward as $V^{\pi}(s)$, which represents the expected return obtained by following policy π from state s, mathematically expressed as:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|S_t = s] \tag{19}$$

And we express the state–action value function of the reward as $Q^{\pi}(s,a)$, which represents the expected return obtained by taking action a in the current state s while following policy π, mathematically

expressed as:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \tag{20}$$

In Eqs. (19) and (20), G_t represents the return, which is the sum of all discounted rewards from the state s_t at time t until the terminal state, given by Eq. (21).

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k} \tag{21}$$

Eq. (21) defines the cumulative discounted return $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$, which is an empirical estimate of the action-value function $Q^{\pi}(s_t, a_t)$ under policy π. In Markov Decision Processes, the Bellman optimality equation for the optimal action-value function Q^* is given by:

$$Q^*(s,a) = \mathbb{E}\left[R(s,a) + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$$

(Bellman Optimality Equation)

Here, G_t represents a sample path of $Q^{\pi}(s_t, a_t)$, approximating the expectation in the Bellman equation with empirical trajectories (Sutton et al., 1998).

Using RL, we first need to model the problem as a Markov decision problem, and in our work, we construct the following MDP problem as $M = (S, A, P, r, \gamma)$, where

- State Space (S): The state space at the end of each day in the transportation network forms a continuous state space, including but not limited to the inventory upper and lower limits of refinery nodes and distribute nodes, remaining processing capacity of refineries, and demand for refined oil products. These parameters collectively describe the specific operational conditions of the supply chain on a daily basis.
- Action Space (A): The action space is also continuous, primarily consisting of the target inventories of refinery nodes and distribute nodes. By adjusting these target inventories, the system can search for optimal solutions in a multi-dimensional decision-making space to achieve overall supply chain optimization.
- State Transition Function (P): The state transition function characterizes how the system evolves from one inventory state to another, modeled as a conditional probability distribution P. Given the inherent uncertainty in market demand — especially at the provincial store nodes — the transition dynamics incorporate this variability, resulting in stochastic state transitions that depend on both the current state and the RL agent's action.
- Reward Function (r): The reward function is designed to align with the operational objectives of minimizing inventory warnings while guiding the system state toward the target inventory levels specified by the upper-level decision-making module. To achieve this, we define the reward function as Eq. (22):

$$r_t = -k_1 O_t - k_2 G_t - k_3 W_t - \alpha V_t \tag{22}$$

where:

  - O_t: Inventory warning penalty, imposed when inventory levels approach their limits.
  - G_t: Loss incurred from discarding excess inventory.
  - W_t: Costs associated with inventory management and transportation.
  - V_t: Feasibility penalty if the current state deviates from the strategy layer's feasible region.

This reward different form the objective of the mathematical programming module (Eq. (17)), but they share the same high-level goal: minimizing transportation cost and inventory deviation while ensuring operational feasibility.
Based on the relative importance of each component in real-world operations, the coefficients are set as follows: $k_1 = 3$, $k_2 = 5$, $k_3 = 1$, and $\alpha = 4$. The highest weight is assigned to G_t, reflecting the critical impact of inventory waste on both cost and supply efficiency. The feasibility penalty V_t is also given a high coefficient to ensure that the agent adheres closely to the strategic decisions made at the upper level. Meanwhile, O_t is penalized moderately to reduce the frequency of inventory warnings without overly constraining the policy exploration. Lastly, W_t carries the lowest penalty, as it mainly reflects routine operational costs.
- Discount factor (γ): the discount factor is set to $\gamma = 0.99$, reflecting the long-term planning nature of the downstream scheduling task. A high discount factor ensures that the agent takes future rewards into account when making decisions, which is crucial for achieving optimal performance in scenarios with delayed feedback.

At the beginning of each episode of RL training, the system first initializes the environment. The agent then obtains the current state and signal from the environment. The state is composed of information returned by four types of nodes in the downstream supply chain, which is converted into vector form to facilitate rapid calculation and decision-making by the agent. Table 2 shows the observation of each type of node. The signal represents an inventory warning signal; when its value is 1, it indicates that the inventory of a certain node has exceeded the preset safety range, prompting the agent to take measures to avoid supply chain interruptions or excessive inventory issues.

Actions are defined as the current inventory of each distribute node and refinery node's storage capacities and processing budgets. The RL layer retrieves current inventory data from the simulator and uses feedback penalties from the strategy optimization layer and other information provided by the simulator to evaluate output performance. The percentage of crude oil and refined oil inventories at each node per day serves as action inputs to determine target inventory levels.

4.2.1. Proximal policy optimization

PPO is a model-free reinforcement learning algorithm that balances training stability and convergence efficiency by constraining policy update magnitude. This study employs the PPO-Clip variant (Schulman et al., 2017), whose core objective function is defined as:

$$L^{PPO}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right) - \alpha V(s_t) + \beta S(\pi_\theta)\right] \tag{23}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ denotes the policy ratio, $\hat{A}_t$ is the Generalized Advantage Estimation (GAE), $V(s_t)$ is the state-value function, $S(\pi_\theta)$ is the policy entropy, and ϵ, α, β are hyperparameters.

- Gaussian Policy Network (GaussianActor): The policy network is a three-layer fully-connected neural network that parameterizes a Gaussian distribution for continuous actions:
  - Input Layer: Maps the state space $\mathbb{R}^{d_s}$ to a hidden layer via an orthogonally initialized linear layer.
  - Hidden Layers: Two ReLU-activated layers with dimension h to capture state features.
  - Output Layers:
    * Mean $\mu(s) \in \mathbb{R}^{d_a}$ via a linear layer with orthogonal initialization (std = 0.01);
    * Log-standard deviation $\log \sigma \in \mathbb{R}^{d_a}$ as trainable parameters (initialized to 0.1).

Action sampling uses the reparameterization trick:

$$a \sim \mathcal{N}(\mu(s), \sigma^2 I), \quad \sigma = \exp(\log \sigma) \tag{24}$$

Table 2
Observation of each type of node.

Node type	Observation
Refinery nodes	Node code, left processing budget, current inventory, storage capacity
Transfer nodes	Node code, current inventory, storage capacity
Provincial library nodes	Node code, current inventory, storage capacity
Sale nodes	Demand

For bounded action spaces $[a_{\min}, a_{\max}]$, actions are clipped via hyperbolic tangent:

$$a_{\text{clip}} = \tanh(a) \cdot \frac{a_{\max} - a_{\min}}{2} + \frac{a_{\max} + a_{\min}}{2} \tag{25}$$

- State-Value Network (Critic): The value network is a three-layer MLP that estimates state values $V(s)$:
  Input layer $\mathbb{R}^{d_s}$ → hidden layers (ReLU) → output layer $\mathbb{R}$, with orthogonal initialization for all layers.
- Experience Collection and Advantage Estimation:
  Rolling Window Sampling: Trajectories $(s_t, a_t, r_t, s_{t+1}, \text{done})$ are stored in a replay buffer with attributes including actions, log probabilities, and values. Generalized Advantage Estimation (GAE):

$$\hat{A}_t = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k}, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{26}$$

  where γ is the discount factor, λ is the GAE parameter. Advantages are normalized to stabilize training:

$$\hat{A}_t \leftarrow \frac{\hat{A}_t - \text{mean}(\hat{A})}{\text{std}(\hat{A}) + \epsilon} \tag{27}$$

- Optimization Procedure:
  Multi-Episode Mini-Batch Updates: Each batch of data is updated for update_freq times with batch size batch_size.
  Gradient Clipping: Parameter gradients are constrained by $\|\nabla\theta\| \leq$ max_grad_norm.
  Learning Rate Decay: Linear decay strategy:

$$\text{lr} = \text{lr}_0 \cdot \left(1 - \frac{\text{episode}}{\text{total\_episodes}}\right) \tag{28}$$

  Loss Function Weighting: Balances policy loss, value loss, and entropy regularization:

$$\text{Loss} = L^{PPO}(\theta) + \text{val\_coef} \cdot L^{VF} - \text{ent\_coef} \cdot S(\pi_\theta) \tag{29}$$

- Algorithm Pseudo-code: The pseudo code of PPO is given in Algorithm 1

Algorithm 1 Proximal Policy Optimization (PPO)

Input: d_s (state dim), d_a (action dim), h (hidden size), hyperparameters
Output: Trained actor π_θ, critic V_ϕ

Initialize:
  Actor π_θ, Critic V_ϕ (orthogonal init.)
  Adam optimizers for θ (actor) and ϕ (critic)
  Replay buffer $\mathcal{B}$
while not converged **do**
  Collect Trajectories: Interact with environment, store $(s_t, a_t, r_t, s_{t+1}, \text{done}) \in \mathcal{B}$
  Preprocess:
    Extract $\{\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{v}, \log\pi, \mathbf{m}\}$ from $\mathcal{B}$
    Compute {returns, advantages} via GAE
    Normalize advantages
  for episode = 1 **to** update_freq **do**
    for mini-batch ∈ BatchSampler(**s**) **do**
      $\log\pi_{\text{new}} \leftarrow \pi_\theta(\mathbf{a}|\mathbf{s})$
      $\text{ratio} \leftarrow \exp(\log\pi_{\text{new}} - \log\pi)$
      $\text{surr1} \leftarrow \text{ratio} \odot \text{advantages}$
      $\text{surr2} \leftarrow \text{clip}(\text{ratio}, 1-\epsilon, 1+\epsilon) \odot \text{advantages}$
      $L_{\text{actor}} \leftarrow -\text{mean}(\min(\text{surr1}, \text{surr2}))$
      $\mathbf{v}_{\text{pred}} \leftarrow V_\phi(\mathbf{s})$
      $\text{returns}_{\text{norm}} \leftarrow \text{returns}/(\text{std}(\text{returns}) + \epsilon)$
      $L_{\text{critic}} \leftarrow \text{MSE}(\mathbf{v}_{\text{pred}}, \text{returns}_{\text{norm}})$
      $L_{\text{entropy}} \leftarrow \text{mean}(-\exp(\log\pi_{\text{new}}) \odot \log\pi_{\text{new}})$
      $\text{Loss} \leftarrow L_{\text{actor}} + \text{val\_coef} \cdot L_{\text{critic}} - \text{ent\_coef} \cdot L_{\text{entropy}}$
      Update: Backpropagate Loss, update θ, ϕ via Adam
        Clip gradients by max_grad_norm
    end for
  end for
  Adjust Learning Rates: Decay $\text{lr}_{\text{actor}}, \text{lr}_{\text{critic}}$ linearly
end while

4.2.2. Soft actor-critic

SAC is an off-policy (Haarnoja et al., 2018) reinforcement learning algorithm that maximizes both expected return and policy entropy to balance exploration and exploitation. It extends actor-critic frameworks with stochastic policies and entropy regularization, making it well-suited for continuous control tasks.

- SAC optimizes three objectives
  Critic Loss (temporal consistency of Q - values):

$$\mathcal{L}_Q = \mathbb{E}_{s,a,r,s'}\left[\frac{1}{2}\left(Q_1(s,a) - y\right)^2 + \frac{1}{2}\left(Q_2(s,a) - y\right)^2\right] \tag{30}$$

  where the target $y = r + \gamma(1-d)\min(Q_1^{\text{target}}(s',a'), Q_2^{\text{target}}(s',a')) - \alpha\log\pi(a'|s')$.
  Actor Loss (policy improvement with entropy regularization):

$$\mathcal{L}_\pi = \mathbb{E}_{s,a}\left[\alpha\log\pi(a|s) - \min(Q_1(s,a), Q_2(s,a))\right] \tag{31}$$

  Entropy Regularization (automated via temperature parameter α):

$$\mathcal{L}_\alpha = \mathbb{E}_{s,a}\left[-\alpha\left(\log\pi(a|s) + \mathcal{H}\right)\right] \tag{32}$$

  where $\mathcal{H}$ is the target entropy (typically $-\dim(\mathcal{A})$).

- Stochastic Policy Network (`GaussianActor`)
  A 3 - layer MLP parameterizing a Gaussian action distribution:
  Input: State $s \in \mathbb{R}^{d_s}$.
  Hidden Layers: Two ReLU - activated layers (dimension h).
  Output:
  - Mean $\mu(s) \in \mathbb{R}^{d_a}$,
  - Log - std $\log\sigma(s) \in \mathbb{R}^{d_a}$ (clamped to $[-20, 2]$ for stability).

  Action sampling uses the reparameterization trick:

$$a \sim \mathcal{N}(\mu(s), \sigma^2 I), \quad \sigma = \exp(\log\sigma(s)) \tag{33}$$

  For bounded action spaces $[a_{\min}, a_{\max}]$, actions are normalized via:

$$a_{\text{clip}} = \tanh(a) \cdot \frac{a_{\max} - a_{\min}}{2} + \frac{a_{\max} + a_{\min}}{2} \tag{34}$$

- Twin Q - Networks (`Critic`)
  Two identical 3 - layer MLPs estimate state - action values $Q_1(s,a)$ and $Q_2(s,a)$:
  - Input: Concatenated state - action pair $(s,a) \in \mathbb{R}^{d_s+d_a}$.

  - Hidden Layers: ReLU - activated layers (dimension h).
  - Output: Scalar Q - value $Q(s,a) \in \mathbb{R}$.

Target Q - networks $Q_1^{\text{target}}, Q_2^{\text{target}}$ are updated slowly via polyak averaging:

$$\theta^{\text{target}} \leftarrow \tau\theta + (1-\tau)\theta^{\text{target}} \tag{35}$$

- Training Pipeline–Experience Replay
  Trajectories $(s_t, a_t, r_t, s_{t+1}, d_t)$ are stored in a replay buffer $\mathcal{B}$, enabling off - policy learning.
- Training Pipeline–Optimization Steps

1. Sample Mini - Batches: Randomly sample batch_size transitions from $\mathcal{B}$.
2. Update Critics:
   - Compute target Q - values y using target networks and policy entropy.
   - Minimize MSE loss between predicted and target Q - values.
3. Update Actor:
   Maximize the objective $\min(Q_1, Q_2) - \alpha \log \pi$ to improve the policy.
4. Update Entropy Temperature α:
   If tune_entropy = True, optimize α to match the target entropy.
5. Update Target Networks:
   Periodically update $Q_1^{\text{target}}, Q_2^{\text{target}}$ via polyak averaging.

- Algorithm Pseudo - Code: The pseudo code of SAC is given in Algorithm 2

Algorithm 2 Soft Actor-Critic (SAC)

Input: $d_s, d_a, h, \gamma, \tau, \alpha, \mathcal{H}$ **Output:** Trained policy π_θ, Q-networks Q_1, Q_2

Initialize:
  Policy π_θ, Q-networks Q_1, Q_2, target Q-networks $Q_1^{\text{target}}, Q_2^{\text{target}}$
  Adam optimizers for $\theta, \phi_1, \phi_2, \alpha$ (if tuned)
  Replay buffer $\mathcal{B}$
while not converged **do**
  Collect Trajectories:
    Interact with environment, store $(s_t, a_t, r_t, s_{t+1}, d_t) \in \mathcal{B}$
  Sample Mini-Batch:
    $(s, a, r, s', d) \sim \text{BatchSampler}(\mathcal{B})$
  Update Critics:
    $a' \sim \pi_\theta(s')$, $\log \pi' \leftarrow \pi_\theta.\text{log\_prob}(a'|s')$
    $y \leftarrow r + \gamma(1-d)\min(Q_1^{\text{target}}(s', a'),\ Q_2^{\text{target}}(s', a')) - \alpha \log \pi'$
    $\mathcal{L}_{Q1} \leftarrow \text{MSE}(Q_1(s,a), y)$, $\mathcal{L}_{Q2} \leftarrow \text{MSE}(Q_2(s,a), y)$
    Update Q_1, Q_2 via backpropagation
  Update Actor:
    $a \sim \pi_\theta(s)$, $\log \pi \leftarrow \pi_\theta.\text{log\_prob}(a|s)$
    $\mathcal{L}_\pi \leftarrow \alpha \log \pi - \min(Q_1(s,a),\ Q_2(s,a))$
    Update π_θ via backpropagation
  Update Entropy Temperature:
  if tune_entropy **then**
    $\mathcal{L}_\alpha \leftarrow -\alpha(\log \pi + \mathcal{H})$
    Update α via backpropagation
  end if
  Update Target Networks:
    $Q_1^{\text{target}} \leftarrow \tau Q_1 + (1-\tau)Q_1^{\text{target}}$
    $Q_2^{\text{target}} \leftarrow \tau Q_2 + (1-\tau)Q_2^{\text{target}}$
end while

4.3. Bilevel rolling-horizon method

The RH method, also known as the Moving Time Domain Technique or the Sliding Window Technique, is a widely adopted strategy in the field of dynamic optimization and control (Sahin et al., 2013). It is primarily used to solve complex dynamic decision-making problems, especially those involving long-term planning and uncertainties. The core idea of this approach is to divide the entire planning period into a series of successive short periods, the so-called "time domains" or "windows", and optimize them for the current time domain. As time progresses, this time domain rolls forward like a wheel, enabling a gradual approach to the long-term goal.

Since the supply chain involves a long time span and complex solution requirements, long-term scheduling problems need to be considered to achieve adaptability in different task scenarios and improve the overall robustness of the algorithm. If the decision-making of transportation volume and inventory volume is made directly on a daily or two-day basis, the training effect will be poor. To this end, we propose a bilevel RH method that makes the use of the advantages of both MP and RL (as shown in Fig. 4). This method splits the long-term planning problem originally measured in months into optimization subproblems in weeks. Specifically, the bilevel RH method consists of a High-Level planning module and a Low-Level scheduling module. The High-Level module employs a RL algorithm to determine the target inventory level for each node, based on the current state and historical data. The current state includes factors such as the storage capacity, storage limits, and demand of the node, while historical data include past reward values that inform long-term decision-making. The Low-Level module optimizes the actual inventory via the precise MP solver, based on the target inventory provided by the High-Level module with RH. On the premise of ensuring the minimum number of early warnings, the actual inventory of the node is as close as possible to the target inventory, minimizing the penalty caused by the upper layer decision-making.

In our framework, the original long-term planning problem formulated in monthly time units is decomposed into a series of short-term optimization sub-problems, each operating on a weekly basis. Within each RL episode, a full planning cycle has to be executed. During this cycle, the rolling window keeps advancing until the entire planning horizon is covered (as demonstrated by iteration n in Fig. 4). Once this process finishes, the next RL episode commences.

At each iteration, the RH method is applied to each weekly subproblem. Specifically, after solving the optimization for a given week, only the decision variables corresponding to the first day are implemented and become part of the final scheduling plan (as illustrated by the blue segment labeled "itr 2" in Fig. 4). Then continue to optimize the next weekly period from the next day until the last week of the month, at which time the optimization plan for the last week is directly output. The entire process went through 24 optimization steps, ensuring a smooth optimization process and high solution quality.

5. Experiment

In the experiment progress, we built a mixed integer linear programming model based on real data, which includes 14 refineries, 19 transfer nodes, 77 provincial library nodes, and 1014 transportation edges. The decision variables of each node and edge are detailed in Table 1. The reinforcement learning environment built based on this model has a 153-dimensional observation space and a 138-dimensional action space, which is a high-dimensional complex model. All experimental schemes are carried out under the premise of keeping this environment unchanged.

During the experiment, we used four different methods to evaluate the effectiveness of our downstream oil supply chain scheduling approach and conducted multiple experiments based on real industrial scenario data. First, the MP solver Gurobi was used to directly solve

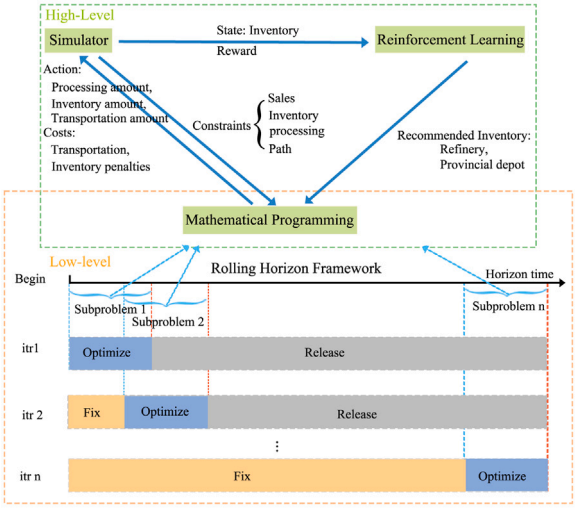

Fig. 4. The bilevel Rolling-Horizon method. The High-level planning module is based on reinforcement learning, while the Low-Level scheduling module is based on mathematical programming solving.

the problem. Secondly, two RL-based methods were introduced: PPO and SAC. These two methods are respectively combined with the MP level to form different optimization frameworks (named as PPO-MP, SAC-MP). Finally, a RH framework was added to improve the quality and adaptability of the solver by dynamically adjusting the decision window, for the framework (named as PPO-MP-RH, SAC-MP-RH).

In the simulation process, the reward of the RL model used to evaluate a solution is based on the cumulative sum of the rewards obtained from the strategies selected at each iteration. Specifically, when conducting experiments using the RH method, the nature of the approach requires re-planning the actions for the next n steps at each decision point, rather than solving the entire problem in one go. As a result, within a single solution trajectory, the solver must be invoked multiple times to update the optimization results for future states. To compute the average reward, we sum all the individual reward values obtained throughout the simulation and divide this total by the number of days involved in the calculation. This yields the average daily reward over the 30-day period.

After adding the RH method, only the time window of the first few days was planned in the initial stage, and the subsequent time periods lacked clear planning. The unplanned time periods triggered more inventory warnings, abandoned losses or other cost penalties, which greatly reduced the overall reward value. It is worth noting that the reward values shown in the experimental results are negative, which stems from the design of the objective function described in Section 4.2. In this formulation, the model aims to minimize penalty terms such as inventory warning penalties, Loss incurred from discarding excess inventory, i.e. . These penalties are represented as negative rewards in the RL framework. Therefore, the RL agent's goal is to maximize the (negative) reward, to reduce the total penalty by making better scheduling decisions over time. As shown in Fig. 5, Even after the introduction of the RH method, the SAC algorithm is still able to maintain a relatively high reward value. This shows that the SAC algorithm has better adaptability and robustness, and can make more effective decisions under incomplete information or partial planning.

In Section 3.4, we set the objective function to minimize the linear sum of the transportation costs of all oil products and the difference between the actual inventory and the target inventory. In order to avoid overfitting, we selected the number of warnings and the solving time as the key indicators to evaluate the performance of the solver.

The changes in the number of warnings of the four different frameworks during the supply chain operation are shown in Fig. 6. The horizontal axis represents the supply chain operation step (in days), while the vertical axis represents the number of warnings. It can be clearly seen from the figure that the number of warnings is significantly reduced after the introduction of the RH method. At the same time, comparing the performances of the frameworks using PPO and the frameworks using SAC, the SAC group shows better performance. This is mainly because the RL algorithm SAC achieves a better balance between exploration and utilization, and has higher sample efficiency and stronger stability, especially when dealing with complex dynamic environments and continuous action space problems. These characteristics enable SAC to more effectively optimize oil transportation costs and inventory management.

Table 3
Comparison of solving time of different frameworks.

Method	Without RH		With RH	
	PPO-MP	SAC-MP	PPO-MP-RH	SAC-MP-RH
Solving time (s)	6.76	6.06	58.60	56.36

To enhance the comprehensiveness of our evaluation, we have incorporated a non-RL baseline method, MP, for a more extensive comparison. At the initial stage, the warning count for MP was zero, as the starting inventory levels sufficed to meet immediate demands without triggering any alerts. In the absence of RL to make preliminary, adaptive decisions, the system relied solely on the static nature of the MP approach. However, as time elapsed and the supply chain environment evolved dynamically, the inflexibility of the MP method became apparent. Without the ability to learn from and respond to changing conditions in real-time, MP gradually accumulated an increasing number of warnings, underscoring its limitations in adapting to the fluid and unpredictable dynamics of the supply chain.

Table 3 provides a critical comparison of total solving time across the full planning horizon. Although the solving time of the frameworks with RH is much longer than those without RH, considering that frameworks with RH solves the original problem by splitting it into multiple sub-problems, the total complexity are naturally higher than those without RH. Therefore, this extra time consumption is reasonable, and the effectiveness and superiority of the RH frameworks are fully verified from the optimization results, especially in reducing the number of warnings. This shows that although the RH method requires more computing resources, it provides a more effective and flexible solution when dealing with complex and dynamic environments. These results further support our exploration of applying advanced algorithms and technologies in the field of industrial automation.

To comprehensively evaluate the impact of the rolling window size on system performance, we analyzed both the number of inventory warnings and the reward curves across different window sizes — namely 5, 7, 9, and 15 — for both PPO-MP-RH and SAC-MP-RH. As shown in Fig. 7, the performance of both algorithms varies significantly with the choice of window size. Specifically, Figs. 7(a) and 7(b) depict the warning count results for PPO and SAC, respectively; while Figs. 7(c) and 7(d) illustrate the corresponding reward curves.

From the warning count analysis, it is evident that the SAC-based method generally outperforms the PPO-based method. Only when using a window size of 5 does a small number of warnings appear in the second half of the simulation period for the SAC algorithm. For PPO:

- Window Size 5: Exhibits significant fluctuations during the early and middle stages, making it suitable for environments requiring rapid adaptation to short-term changes.
- Window Size 7: Shows an initial transient peak but subsequently performs most stably, making it ideal for applications where long-term stability is crucial.
- Window Size 9: Demonstrates large fluctuations during the early stages, along with noticeable variations on certain days; thus, it is appropriate for situations sensitive to short-term changes but capable of tolerating some instability.

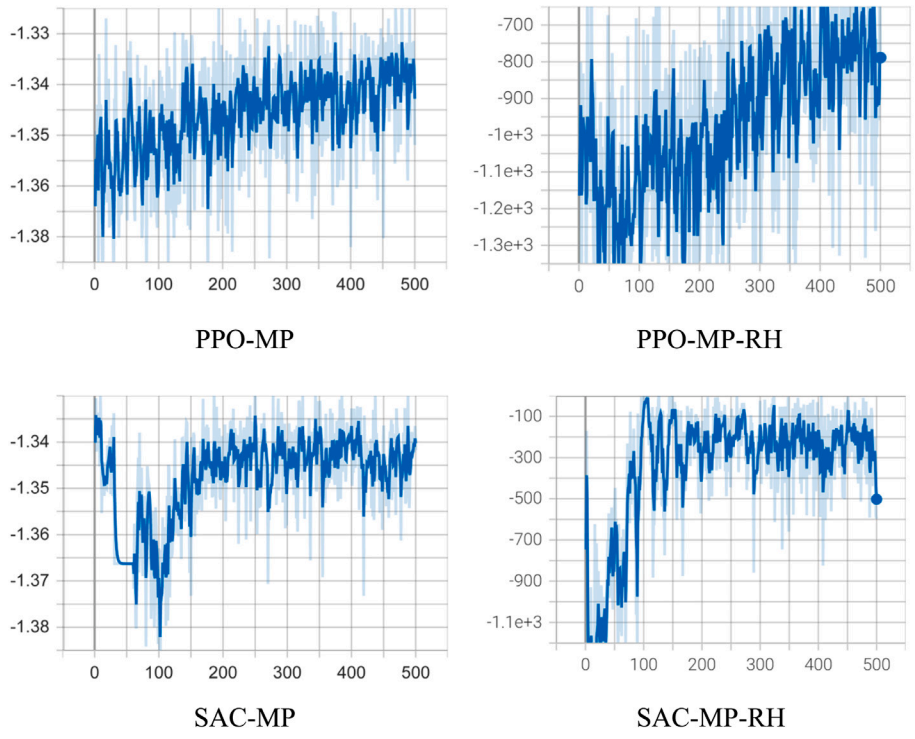

Fig. 5. Training rewards of episodes of different frameworks.

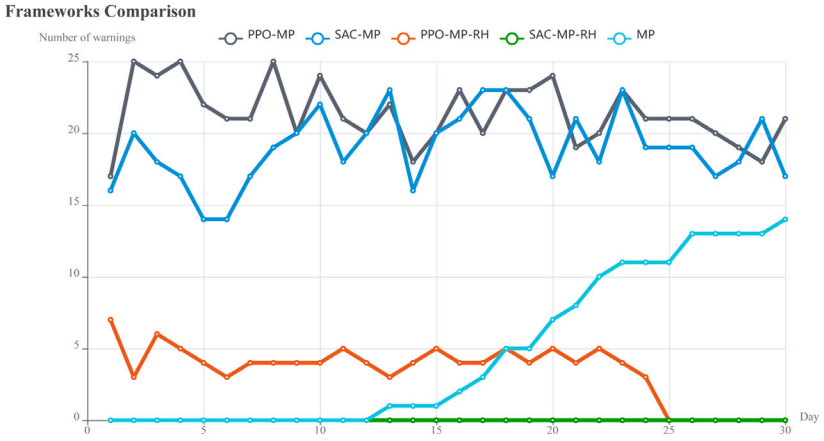

Fig. 6. Comparison of the number of warnings by different frameworks.

- Window Size 15: Displays exceptional stability, with almost no warnings generated after the 13th day, making it best suited for environments where minimizing warning frequency and ensuring high stability are top priorities.

Turning to the reward curves (Figs. 7(c) and 7(d)), similar trends are observed. In general, higher stability in warning counts correlates with smoother and higher cumulative rewards, especially for the SAC algorithm. Notably:

- Window Size 5: While showing more fluctuation in both warning count and reward, this setting enables faster adaptation to dynamic changes, albeit at the cost of increased instability.
- Window Size 7: Offers a good balance between responsiveness and reward consistency, particularly under the SAC algorithm.
- Window Size 9: Although it yields moderate warning control, its reward pattern indicates sensitivity to environmental variability.
- Window Size 15: Provides the smoothest reward curve and the least number of warnings, reinforcing its suitability for stable and predictable environments.

6. Conclusions and future work

This paper proposes a bilevel scheduling approach, which aims to optimize the operations of downstream oil supply chain, including refinery production scheduling, logistics distribution and inventory management. By combining RL with traditional MP, the approach can effectively cope with complex and dynamically changing scheduling problems. A real-data-based simulator is employed to validate the algorithm's effectiveness, and a bilevel RH method is introduced to promptly address immediate needs while preserving the stability of long-term planning. Experimental results show that the RL algorithm SAC with bilevel RH performs particularly well in dealing with complex

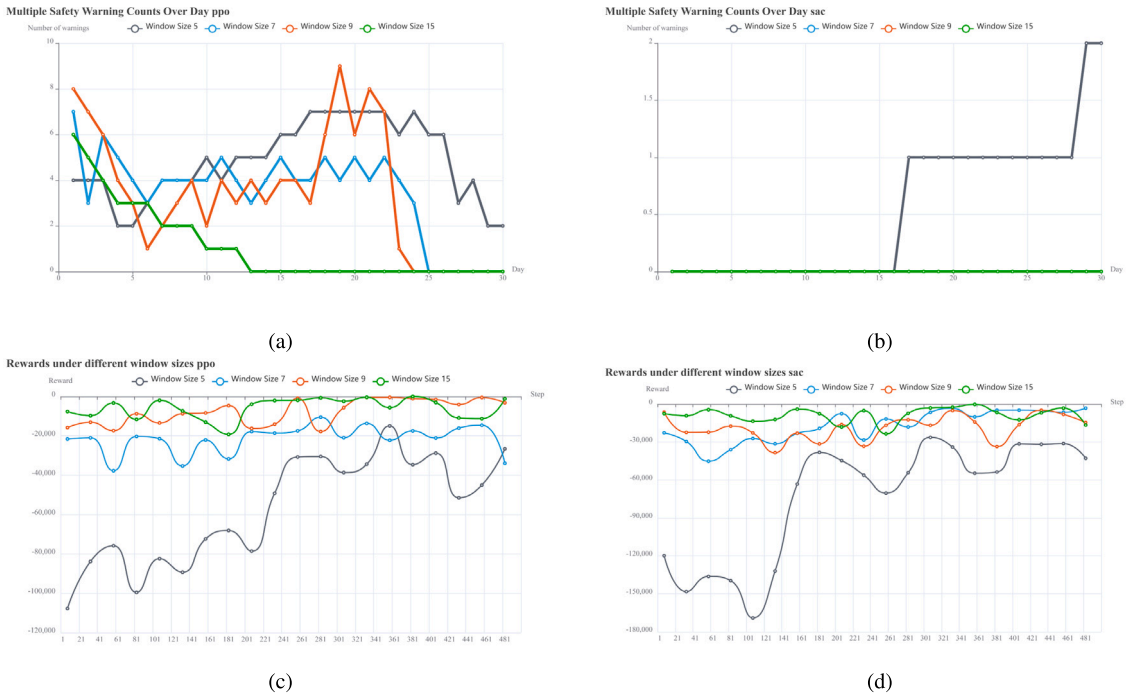

Fig. 7. Comparison of different window sizes for (a, c) PPO-MP-RH and (b, d) SAC-MP-RH.

dynamic environments and continuous action space problems, significantly reducing the number of warnings and improving the overall optimization results.

In future work, the following aspects will be considered:

- In the oil supply chain, each node can be regarded as an independent agent. These agents need to work together to optimize the entire supply chain. Through Multi-Agent RL, each agent can make decisions based on its own state and goals, and interact and coordinate with other agents to achieve the global optimal or near-optimal scheduling solution.
- In addition, due to the complex interactions between the nodes, a directed acyclic graph is formed. In order to accurately capture these complex interactions, we can introduce a framework based on Graph Neural Networks. This framework enables agents to effectively understand and process information transfer in the network structure, thereby improving the performance of the overall system.
- Furthermore, to better evaluate the advantages of the proposed bilevel RL approach, a comparative study will be conducted between heuristic methods (such as genetic algorithms and simulated annealing) and advanced optimization techniques like multi-parametric programming. This comparison will help identify the trade-offs between computational efficiency, solution quality, and adaptability to dynamic environments, providing insights into the most suitable method for different operational scenarios.

During the preparation of this work the author used Qwen in order to polish this article. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication.

CRediT authorship contribution statement

Qipeng Yang: Writing – original draft, Validation, Investigation, Formal analysis. **Wentian Fan:** Visualization, Data curation. **Nan Ma:** Supervision, Resources, Funding acquisition. **Shu Lin:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization. **Jiawen Chang:** Supervision. **Zhiqiang Zou:** Writing – review & editing, Supervision. **Liang Sun:** Supervision. **Haifeng Zhang:** Supervision, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Shu Lin reports financial support was provided by Petrochina Planning and Engineering Institute. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by Petrochina Planning & Engineering Institute for the project “Research on Key Technologies and Product Development for Intelligent Scheduling of Oil and Gas Industry Chain Production and Operation” (Project No.: 2023DJ7708).

Data availability

The authors do not have permission to share data.

References

Abdolazimi, O., Abraham, A., 2021. Meta-heuristic based multi objective supply chain model for the oil industry in conditions of uncertainty. In: Innovations in Bio-Inspired Computing and Applications: Proceedings of the 11th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2020) Held During December 16-18, 2020 11. Springer, pp. 141–153.

Adhitya, A., Srinivasan, R., Karimi, I.A., 2007. Heuristic rescheduling of crude oil operations to manage abnormal supply chain events. AIChE J. 53 (2), 397–422.

Al-Othman, W.B., Lababidi, H.M., Alatiqi, I.M., Al-Shayji, K., 2008. Supply chain optimization of petroleum organization under uncertainty in market demands and prices. European J. Oper. Res. 189 (3), 822–840.

Bányai, T., Veres, P., Illés, B., 2015. Heuristic supply chain optimization of networked maintenance companies. Procedia Eng. 100, 46–55.

Cplex, I.I., 2009. V12. 1: User's manual for CPLEX. Int. Bus. Mach. Corp. 46 (53), 157.
Dai, X., Zhao, L., Li, Z., Du, W., Zhong, W., He, R., Qian, F., 2021. A data-driven approach for crude oil scheduling optimization under product yield uncertainty. Chem. Eng. Sci. 246, 116971.
Dell'Aversana, P., 2024. Oil production optimization based on reinforcement learning and self-aware deep neural networks. In: 85th EAGE Annual Conference & Exhibition (Including the Workshop Programme). 2024, (1), European Association of Geoscientists & Engineers, pp. 1–5.
Göthe-Lundgren, M., Lundgren, J.T., Persson, J.A., 2002. An optimization model for refinery production scheduling. Int. J. Prod. Econ. 78 (3), 255–270.
Gurobi Optimization, L., 2024. Gurobi Optimizer Reference Manual. URL: https://www.gurobi.com.
Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al., 2018. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.
Hou, Y., Wu, N., Zhou, M., Li, Z., 2015. Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm. IEEE Trans. Syst. Man, Cybern.: Syst. 47 (3), 517–530.
Hussain, R., Assavapokee, T., Khumawala, B., 2006. Supply chain management in the petroleum industry: challenges and opportunities. Int. J. Glob. Logist. & Supply Chain Manag. 1 (2), 90–97.
International Energy Agency (IEA), 2024. Oil market report - April 2024. Licence: CC BY 4.0 https://www.iea.org/reports/oil-market-report-april-2024.
Joly, M., Moro, L.F.L., Pinto, J.M., 2002. Planning and scheduling for petroleum refineries using mathematical programming. Braz. J. Chem. Eng. 19, 207–228.
Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement learning: A survey. J. Artificial Intelligence Res. 4, 237–285.
Kelly, J., Mann, J., 2003. Crude oil blend scheduling optimization: an application with multimillion dollar benefits–part 1: The ability to schedule the crude oil blendshop more effectively provides substantial downstream benefits.(special report: Process/plant optimization). Hydrocarb. Process. 82 (6), 47–52.
Kim, S., Choi, Y., Park, J., Adams, D., Heo, S., Lee, J.H., 2024. Multi-period, multi-timescale stochastic optimization model for simultaneous capacity investment and energy management decisions for hybrid micro-grids with green hydrogen production under uncertainty. Renew. Sustain. Energy Rev. 190, 114049.
Konda, V., Tsitsiklis, J., 1999. Actor-critic algorithms. Adv. Neural Inf. Process. Syst. 12.
Li, J., Misener, R., Floudas, C.A., 2012. Scheduling of crude oil operations under demand uncertainty: A robust optimization framework coupled with global optimization. AIChE J. 58 (8), 2373–2396.
Li, F., Yang, M., Du, W., Dai, X., 2020. Development and challenges of planning and scheduling for petroleum and petrochemical production. Front. Eng. Manag. 7 (3), 373–383.
Lisitsa, S., Levina, A., Lepekhin, A., 2019. Supply-chain management in the oil industry. In: E3S Web of Conferences. 110, EDP Sciences, p. 02061.
Makkar, S., Devi, G.N.R., Solanki, V.K., 2020. Applications of machine learning techniques in supply chain optimization. In: ICICCT 2019–System Reliability, Quality Control, Safety, Maintenance and Management: Applications To Electrical, Electronics and Computer Science and Engineering. Springer, pp. 861–869.
Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning. PmLR, pp. 1928–1937.
Moro, L.F., Pinto, J.M., 2004. Mixed-integer programming approach for short-term crude oil scheduling. Ind. Eng. Chem. Res. 43 (1), 85–94.
Mouret, S., Grossmann, I.E., Pestiaux, P., 2011. A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. Comput. Chem. Eng. 35 (12), 2750–2766.
Rolf, B., Jackson, I., Müller, M., Lang, S., Reggelin, T., Ivanov, D., 2023. A review on reinforcement learning algorithms and applications in supply chain management. Int. J. Prod. Res. 61 (20), 7151–7179.
Saharidis, G.K., Minoux, M., Dallery, Y., 2009. Scheduling of loading and unloading of crude oil in a refinery using event-based discrete time formulation. Comput. Chem. Eng. 33 (8), 1413–1426.
Sahebi, H., Nickel, S., Ashayeri, J., 2014. Strategic and tactical mathematical programming models within the crude oil supply chain context—A review. Comput. Chem. Eng. 68, 56–77.
Sahin, F., Narayanan, A., Robinson, E.P., 2013. Rolling horizon planning in supply chains: review, implications and directions for future research. Int. J. Prod. Res. 51 (18), 5413–5436.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
Sutton, R.S., Barto, A.G., et al., 1998. Reinforcement Learning: An Introduction. Vol. 1, MIT press Cambridge.
Zahedi-Seresht, M., Sadeghi Bigham, B., Khosravi, S., Nikpour, H., 2024. Oil production optimization using Q-learning approach. Process. 12 (1), 110.
Zavvari, S., Saifoddin Asl, A., Optimizing oil allocation: a two-stage decision framework for petrochemicals and refineries using deep learning and reinforcement learning. Optimizing Oil Allocation: A Two-Stage Decision Framework for Petrochemicals and Refineries Using Deep Learning and Reinforcement Learning.