mathematics

MDPI

Article

State-Space Compression for Efficient Policy Learning in Crude Oil Scheduling

Nan Ma [1], Hongqi Li [1,*] and Hualin Liu [2,3]

[1] School of Information Science and Engineering, China University of Petroleum, Beijing 102249, China; 2019310420@student.cup.edu.cn
[2] Petrochina Planning and Engineering Institute, Beijing 100083, China; liuhualin08@petrochina.com.cn
[3] Key Laboratory of Oil & Gas Business Chain Optimization, CNPC, Beijing 100083, China
* Correspondence: hq.li@cup.edu.cn

Abstract: The imperative for swift and intelligent decision making in production scheduling has intensified in recent years. Deep reinforcement learning, akin to human cognitive processes, has heralded advancements in complex decision making and has found applicability in the production scheduling domain. Yet, its deployment in industrial settings is marred by large state spaces, protracted training times, and challenging convergence, necessitating a more efficacious approach. Addressing these concerns, this paper introduces an innovative, accelerated deep reinforcement learning framework—VSCS (Variational Autoencoder for State Compression in Soft Actor–Critic). The framework adeptly employs a variational autoencoder (VAE) to condense the expansive high-dimensional state space into a tractable low-dimensional feature space, subsequently leveraging these features to refine policy learning and augment the policy network's performance and training efficacy. Furthermore, a novel methodology to ascertain the optimal dimensionality of these low-dimensional features is presented, integrating feature reconstruction similarity with visual analysis to facilitate informed dimensionality selection. This approach, rigorously validated within the realm of crude oil scheduling, demonstrates significant improvements over traditional methods. Notably, the convergence rate of the proposed VSCS method shows a remarkable increase of 77.5%, coupled with an 89.3% enhancement in the reward and punishment values. Furthermore, this method substantiates the robustness and appropriateness of the chosen feature dimensions.

Keywords: crude oil scheduling; efficient policy learning; state-space compression; reinforcement learning

MSC: 68T05

Citation: Ma, N.; Li, H.; Liu, H. State-Space Compression for Efficient Policy Learning in Crude Oil Scheduling. *Mathematics* **2024**, *12*, 393. https://doi.org/10.3390/math12030393

Academic Editor: Florin Leon, Mircea Hulea and Marius Gavrilescu

Received: 17 December 2023
Revised: 14 January 2024
Accepted: 19 January 2024
Published: 25 January 2024

1. Introduction

The orchestration of crude oil storage and transportation scheduling is pivotal at the forefront of refinery operations, underpinning the safety of oil storage and transit, the stability of production, and the operational efficiency of the refinery [1]. This complex process encompasses the unloading of tankers, the coordination of terminal and factory tank storage, and the seamless transfer of resources to the processing apparatus. Effective scheduling requires intricate decision making across various operational phases, including the timely and precise movement of crude oil to designated units [2]. Objectives focus on maintaining uninterrupted processing, minimizing tanker delays, and optimizing resource allocation across storage and processing units. Operational dispatch must also navigate a myriad of practical considerations, from the punctuality of tanker arrivals to the preparedness of storage facilities and the interconnectivity of various systems. Addressing this large-scale, multiconstraint scheduling challenge is pivotal, representing a dynamic research frontier demanding innovative and efficient solutions.

Contemporary research methodologies addressing refinery crude oil scheduling predominantly draw upon operations research theory [3,4]. These approaches typically entail the formulation of the problem into a mathematical model amenable to solution [5–7]. The strength of this strategy lies in its capacity for the precise mathematical articulation of the scheduling process and production objectives, as well as in its ability to identify provably optimal solutions. However, the timeliness of these solutions poses a significant challenge. Presently, refinery crude oil scheduling is often represented and tackled as a large-scale mixed integer programming model, characterized as an NP-hard problem. Absent simplification, such models defy resolution within a practical timeframe.

Recent advancements in deep reinforcement learning have led to notable successes in tackling complex planning problems [8], prompting numerous research initiatives and applications in the realm of production resource scheduling with promising outcomes [9–12]. This methodology models business challenges as Markov decision processes and learns policies that maximize cumulative rewards through sustained interaction with the environment. Its core strengths lie in its neural-network-based approximation capabilities, rapid sequential decision making, and a degree of adaptability in addressing dynamic programming challenges [13]. Yet, when applied to actual industrial problems, these methods often grapple with expansive state spaces, extended training durations, and convergence difficulties [14], signaling the need for more efficient methods.

This study introduces a novel approach, termed Variational Autoencoder for State Compression in Soft Actor–Critic (VSCS), to model and expedite the training of deep reinforcement learning for refinery scheduling tasks. Initially, this research delineates the Markov decision process for refinery scheduling to lay the groundwork for subsequent optimization. The VSCS methodology employs a variational autoencoder to transmute the extensive, high-dimensional state space into a condensed, low-dimensional representation. Utilizing these distilled features, the VSCS algorithm learns the optimal policies in the reduced feature space, substantially enhancing both the learning efficiency and the efficacy of the derived policies. The paper's principal contributions are multifaceted, encompassing the following key dimensions:

- A novel deep reinforcement learning framework, VSCS, is presented, employing a variational autoencoder to distill the complex, high-dimensional state space of refinery crude oil scheduling into a compact, low-dimensional feature space for optimal policy identification.
- To address the challenge of selecting the dimensionality for low-dimensional features, we devised a method that rigorously evaluates the similarity of feature reconstructions. This approach, integrated with visual analytics, enables the precise determination of the optimal dimensionality for low-dimensional features.
- The VSCS approach delineated herein underwent comprehensive experiments within the crude oil scheduling problem, conclusively affirming the framework's efficacy. Experimental validation confirmed the appropriateness of the chosen low-dimensional feature dimensions, establishing a robust empirical foundation for the methodology.

The remainder of this paper is organized as follows. A brief review of related work is presented in Section 2. Section 3 shows the problem formulation. Section 4 presents the details of the VSCS method. Section 5 delineates and deliberates upon the principal experimental outcomes. Finally, some concluding remarks are given in Section 6.

2. Related Work

Crude oil storage and transportation scheduling are critically important to refinery production. This sequential decision-making process encompasses oil tanker arrival and unloading at the port, the conveyance of crude oil from terminal storage to in-plant tanks, and the subsequent delivery of crude materials to processing units. The overarching objective of scheduling is to minimize the cumulative costs, such as operational expenses, while adhering to the operational capabilities of each segment and maintaining the continuous, planned operation of processing units [15].

Production scheduling presents a multifaceted challenge extensively explored within the mathematical programming sphere, with research bifurcating into modeling methodologies and algorithmic solutions. Shah et al. pioneered a discrete-time Mixed-Integer Linear Programming (MILP) framework to navigate the intricacies of crude oil scheduling [16]. Advancing this groundwork, J.M. Pinto et al. crafted mixed-integer optimization models that capture the dichotomy of continuous and discrete temporal dynamics for refinery scheduling [17]. Jialin Xu's team leveraged continuous-time models for the simulation optimization of refinery operations, showcasing efficacy in scheduling and economic performance [1]. Further refining these approaches, Bernardo Zimberg et al. employed continuous-time models with intricate multioperation sequencing, achieving hourly resolution in their analyses [18]. Lijie Su introduced an innovative continuous–discrete-time hybrid model that stratifies refinery planning and scheduling into hierarchical levels, focusing on multiperiod crude oil scheduling with the aim of maximizing net profits, achieving solution times that range from minutes to hours [19]. Algorithmically, solutions span from MILP-NLP decomposition to solver-integrated responses [20–22] and rolling horizon strategies for time-segmented problem-solving [23]. Additionally, intelligent search mechanisms like genetic algorithms have been adopted to bolster solution throughput [24–27]. Traditional algorithms have thus concentrated on the meticulous detail of model construction and improving efficiency in confronting the complexities of refinery oil storage and transportation. Modeling has progressed from linear representations to intricate nonlinear continuous-time frameworks to mirror operational realities more closely. Nevertheless, the elevated complexity of such models demands the decomposition of problems into tractable subproblems suitable for solver optimization or the application of heuristics and genetic algorithms for more rapid approximate solutions. Consequently, advancing the performance of solutions in this domain remains an ongoing and formidable research challenge. Table 1 shows the different scales and corresponding performances of the calculation examples in the traditional method research of the crude oil scheduling problem.

Table 1. The scale and performance of traditional research methods in crude oil scheduling.

Technique	Scale	Performance
discrete-time MILP framework [16]	Four crude types, two CDUs, seven refinery tanks, and eight portside tanks; the time horizon of operation is one month, and a discretization interval of one day is used	in a few minutes
continuous and discrete temporal MILP [17]	Three CDUs, six storage tanks, and three oil pipelines; the time horizon of operation is one day, at every hour	in reasonable time
continuous-time MINLP [1]	One single docking berth, four storage tanks, four charging tanks, and two CDUs; the time horizon of operation is 15 days	25.94 s
Many-objective optimization for scheduling of crude oil operations based on NSGA-III [26]	There are three distillers with nine charging tanks and a long-distance pipeline; every time, it needs to produce a 10-day schedule	about 100 s–150 s
MILP framework with rolling horizon strategy [23]	Eight tanks, where one tank is assumed in maintenance, five crude qualities; the time horizon is 31 or 61 days (periods)	less than 5 min

Deep reinforcement learning (DRL) has emerged as a potent tool for complex decision-making challenges, with its application broadening significantly in recent years [28]. The method distinguishes itself through formidable learning and sequential decision-making capabilities, facilitating swift, dynamic scheduling decisions in diverse real-world scenarios. In the realm of manufacturing, Christian D. et al. employed DRL in the scheduling of chemical production, adeptly managing uncertainties and facilitating on-the-fly processing decisions, thereby surpassing the performance of MILP models [29]. Yong et al. pioneered a DRL-based methodology for dynamic flexible job-shop scheduling (DFJSP), focused on curtailing average delays through policy network training via the DDPG algorithm, thereby eclipsing rule-based and DQN techniques [30]. Che et al. aimed to curtail total operational

expenditures to minimize energy usage and reduce the frequency of operational mode transitions, enhancing stability. For this, they utilized the PPO algorithm to train decision networks, yielding quantifiable improvements in cost-efficiency and mode-switching [31]. Lee et al. harnessed DRL to orchestrate semiconductor production line scheduling to align with production agendas, selecting DQN as the algorithm of choice and establishing strategies apt for dynamic manufacturing environments [32]. In the transportation field, Yan et al. addressed the intricacies of single-track railway scheduling, which encompasses train timetabling and station track allocation, via a sophisticated deep learning framework, securing superior results in large-scale scenarios in comparison with the commercial solver IBM CPLEX [33]. Furthermore, Pan et al. implemented hierarchical reinforcement pricing predicated on DDPG to solve the intricate distribution puzzles presented by shared bicycle resources, consequently achieving enhancements in service quality and bicycle distribution [34].

The extant research reveals that prevailing reinforcement learning methodologies face constraints in their deployment for large-scale industrial applications. These constraints arise from the considerable scale and intricacy of the scenarios, which give rise to extensive state–action spaces, thus hindering the efficiency of learning processes [14,35,36]. Within the domain of refinery crude oil scheduling, analogous challenges are encountered. To mitigate these challenges, the present study proposes the VSCS framework, which transposes the original, high-dimensional state space into a more compact, lower-dimensional feature space, thereby improving the learning process for the complexities of crude oil scheduling tasks.

3. Problem Formulation

3.1. Description of the Refinery Scheduling Problem

The refinery scheduling problem presented in this paper can be depicted as an operational process, as illustrated in Figure 1. It encompasses the arrival of crude oil tanker V_a at the port for unloading into designated port storage tanks. These tanks include owned storage vessels V_d and commercial storage vessels V_b. Following the desalting and settling operations of crude oil, the port storage tanks can transfer the oil to the in-plant tanks V_f as required via the long-distance pipeline V_p. Terrestrial crude oil V_l enters the in-plant storage tanks through the pipeline. The in-plant tank area is tasked with blending different types $m \in M$ of crude oil according to the processing schemes of the processing units V_u and transporting them to the processing units for refining.

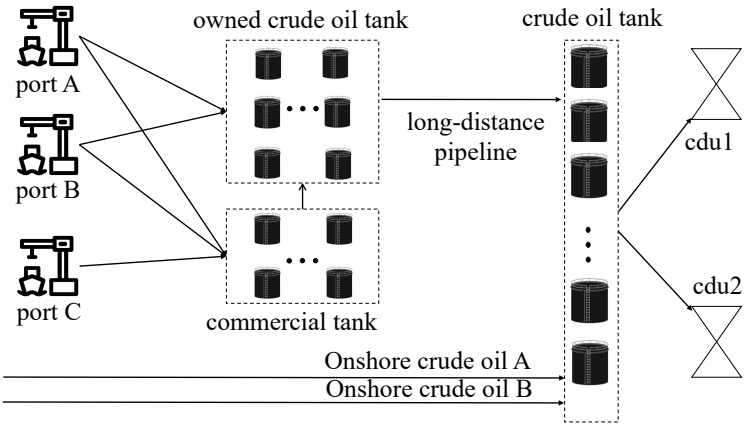

Figure 1. Schematic diagram of refinery crude oil scheduling scenario.

The initial conditions for the scheduling decision process include the anticipated arrival time of oil tankers and the storage tanks projected for unloading, the type of crude oil and the liquid level heights ($L_{vi}^{m,t0}$) stored in each tank at the outset, the upper and lower limits of tank liquid levels (C_{Ub}, C_{Lb}), the upper limit of the long-distance pipeline C_p,

and the topology of the scheduling network. The operational constraints considered are as follows:

- Within a single cycle, each tank must contain only one type of oil product.
- Communal storage tanks and dock tanks can only commence oil transfer operations after completing static desalting.
- The liquid levels in all storage tanks must be maintained within the specified upper and lower capacity limits.
- The transfer rates must remain within the safe transfer speed range.
- Crude oil transported via overland pipelines enters the factory tanks at a predetermined rate.
- The processing units must operate continuously in accordance with the specified processing schemes and plans.

3.2. Markov Modeling

The scheduling objective of this study is to devise a decision-making scheme that minimizes scheduling costs within a short cycle of seven days (with each time step being four hours) while considering the operational constraints of refinery scheduling and the continuity of processing units. The decision scheme includes the oil transfer rates and target tanks for each storage unit. The refinery crude oil scheduling issue can be viewed as a sequential decision-making problem, where the operational process can be described by the fact that the state of each node in the refinery's crude oil storage and transportation operation in the next period is based on the decisions made in the current period, hence the scheduling issue can be modeled as a Markov decision process.

In the refinery scheduling Markov decision process, the type and level of materials in each storage tank are closely related to the scheduling objectives following operational execution. Moreover, the refinery's processing units require continuous feeding according to the processing plan; thus, the remaining processing volumes of various materials in the units must also be considered. Based on these considerations, the state is defined as follows, as illustrated in Equation (1).

$$S = \left\{ S_{va}^{t}, S_{vb}^{t}, S_{vd}^{t}, S_{vp}^{t}, S_{vf}^{t}, S_{vu}^{t} \right\} \tag{1}$$

where S_{va}^{t} includes $L_{va}^{m,t}$, which is the remaining unloading time of the tanker and other attribute information (such as node name). S_{vb}^{t}, S_{vd}^{t}, S_{vf}^{t}, respectively, represent the corresponding tank-level information $L_{vb}^{m,t}$, $L_{vd}^{m,t}$, $L_{vf}^{m,t}$, other attribute information (such as node name), etc. S_{vp}^{t} represents the oil head information connecting the terminal pipe area and the commercial storage pipe area, pipeline transportation volume L_{p}^{t}, etc.; S_{vu}^{t} includes the processing plan and the remaining processing volume of the device.

For the refinery's crude oil storage and transportation scheduling problem, the decision-making network is required to determine the appropriate scheduling actions in response to the varying states at each time period t. The action space is defined by the operational requirements of each node, with the specific action definitions provided in Equation (2).

$$A = \left\{ A_{V_a}^{t}, A_{V_b}^{t}, A_{V_d}^{t}, A_{V_p}^{t}, A_{V_u}^{t} \right\} \tag{2}$$

where $A_{V_a}^{t}$ represents the joint decision-making action of the V_a node, including the oil unloading speed. $A_{V_b}^{t}$, $A_{V_d}^{t}$ are the joint decision-making actions of V_b and V_d, respectively, including the oil payment speed of commercial storage tanks and terminal tank node and the oil payment target node. $A_{V_p}^{t}$ is the pipeline transportation speed, and $A_{V_u}^{t}$ includes processing speed.

In the proposed refinery crude oil scheduling model, each action executed during a scheduling step is assessed by the system through corresponding rewards, which serve to evaluate the efficacy of the action strategy. The objective of this model is to concurrently

minimize operational events and maximize adherence to production constraints, according to the stipulated full-cycle processing plan. To facilitate the agent's strategy enhancement in alignment with this objective during training, the reward function is crafted to precisely guide action decisions. This function is expressed through Equations (3)–(6). Given that the algorithm aims to optimize the long-term average reward, the reward function is structured with negative values that are proportional to associated costs.

$$R = -\omega_0 R_0 - \omega_1 R_1 - \omega_2 R_2 \tag{3}$$

$$R_0 = \sum_{t\in T}\sum_{i\in\{b,d,f\}}\left(O_d \times \left(\left(C_{Ulb_{vi}} - L_{vi}^{m,t}\right) + \left(L_{vi}^{m,t} - C_{Lb_{Vi}}\right)\right)\right) + \sum_{t\in T} O_d \times \left(L_p^t - C_p\right) + \sum_{t>T_a} O_a \times L_{v_a}^{m,t} \tag{4}$$

$$R_1 = O_p \times \left(NP_a + NP_b + NP_d + NP_f\right) + O_b \times \left(Nb_d + Nb_f + \sum_{i\in V_u} Nb_i\right) \tag{5}$$

$$R_2 = \sum_{t\in T} O_c L_{vi}^{m,t} \tag{6}$$

As shown in the above equation, R consists of three parts, where ω is the weight factor of each part; R_0 is the reward and punishment for exceeding the operation constraint, which is composed of each storage tank and the pipeline exceeding the operation constraint and the oil tanker overdue constraint; and R_1 is the speed fluctuation reward and punishment, that is, operation switching. The rewards and punishments are, respectively, composed of the oil tanker speed unloading switching, the oil payment switching of each storage tank, the processing device processing speed switching, and the reward and punishment for the oil type switching. R_2 is the reward and punishment for the inventory cost.

In our model, O_a denotes the cost coefficient associated with the delay in oil tanker unloading, with T_a representing the corresponding delay time. O_p is defined as the cost coefficient for speed fluctuations, while NP_v indicates the number of such fluctuations at each node. The term O_b refers to the cost incurred due to switching between different types of oil, with Nb_v quantifying the frequency of these oil species switches at each node. Lastly, O_d represents the cost coefficient for instances when the liquid level exceeds predetermined upper and lower limits, and O_c signifies the cost coefficient related to inventory management.

4. The Proposed VSCS Algorithm

4.1. The Framework of VSCS

The VSCS framework introduced in this study comprises two primary modules: the low-dimensional feature generation module and the policy learning module. The former autonomously extracts a condensed, low-dimensional feature representation, while the latter module leverages these features to facilitate efficient policy learning. Figure 2 delineates the structural organization and operational sequence of the VSCS framework within the context of refinery crude oil resource scheduling.

As depicted in Figure 2, the policy learning module, rooted in deep reinforcement learning, principally employs the Soft Actor–Critic (SAC) framework. This framework encompasses a policy network, a state value network, and an action value network. The objective is to deduce the appropriate reward feedback following state transitions within the refinery's crude oil storage and transportation scheduling environment. This is achieved by reconstructing the state into a lower-dimensional representation for efficient network training and subsequent action strategy formulation. The state low-dimensional feature generation module functions as a pretraining mechanism, utilizing an encoder network trained via the VAE architecture to transform the state space into a reduced feature space.

This transformation is instrumental in facilitating the strategic training of the main framework. Each module is expounded upon in the subsequent sections.

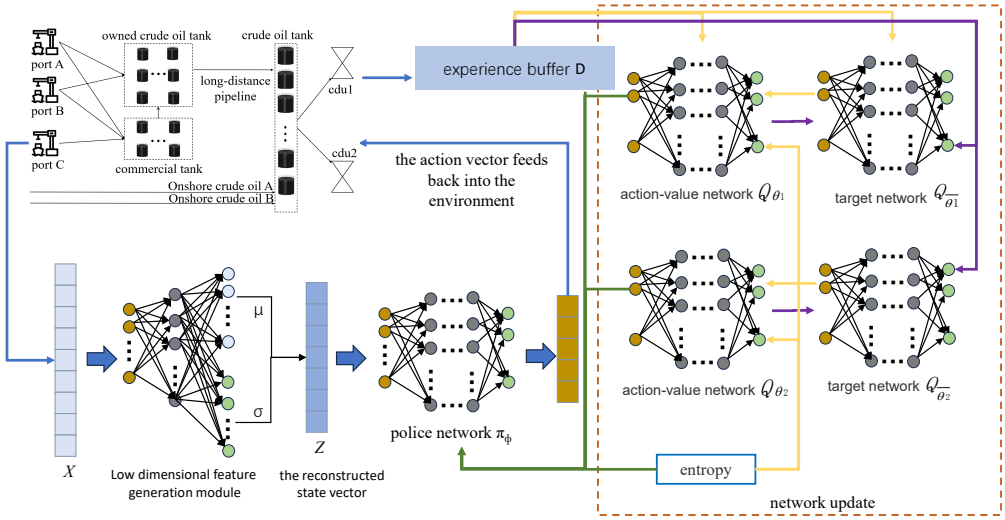

Figure 2. Framework diagram of the proposed VSCS algorithm.

4.2. Low-Dimensional Feature Generation Module

The objective of the low-dimensional feature generation module is to transmute the original, high-dimensional state space into a more tractable, low-dimensional state space while preserving the integrity of the state information to the greatest extent possible. This study employs a VAE to produce low-dimensional state features through unsupervised learning [37]. The VAE operates as a probabilistic model grounded in variational inference, comprising two primary components. The first is the encoder, which is tasked with condensing the high-dimensional state X into a compact, low-dimensional representation Z, which obeys Gaussian distribution and is composed of μ and σ generated by the encoder. The complementary component of the VAE is the decoder, which functions to regenerate the original features by reconstructing the latent variable Z back into the state transition vector X', as illustrated in Figure 3. More computation details are shown in Algorithm 1.

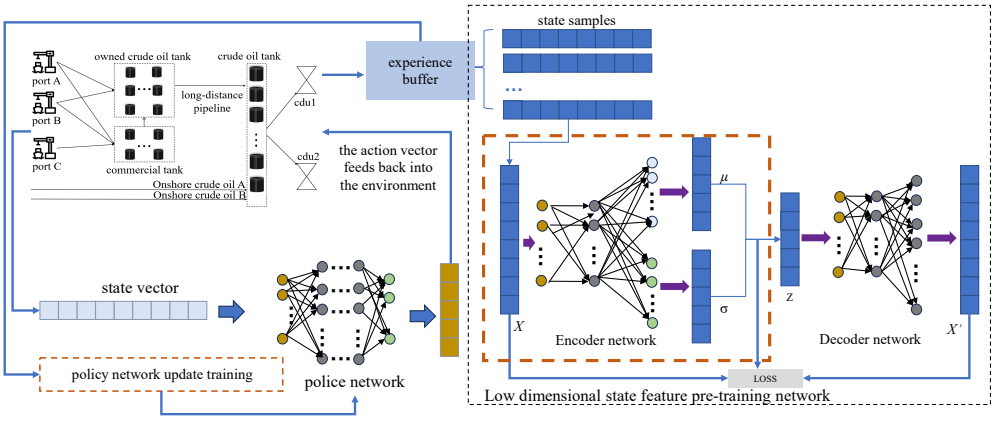

Figure 3. Framework diagram of the low-dimensional feature generation module.

In accordance with Bayesian principles, the joint probability distribution of the observed state vector X and the latent variable Z can be represented as depicted in Equation (7).

$$p(Z \mid X) = p(X \mid Z)p(Z)/p(X) \tag{7}$$

However, due to the intractability of $p(X)$, this study introduces an alternative distribution to approximate $p(Z \mid X)$. This approximative distribution, denoted as $q_\beta(Z \mid X)$, serves as an estimation of the posterior model (encoder), whereby Z is derived from X. The distribution denoted as $p_\eta(X \mid Z)p_\eta(Z)$ corresponds to the generative model (decoder). The encoder and decoder training process involves the concurrent learning of parameters β and η.

A central aspect of this work is the simultaneous training of the approximate posterior model and the generative model by maximizing the variational lower bound, which is articulated in Equation (8).

$$\zeta = -D_{KL}\big(q_\beta(Z \mid X) \| p_\beta(Z)\big) + \mathrm{E}_{q_\beta(Z|X^{(i)})}\Big[\log p_\eta\Big(X^{(i)} \mid Z\Big)\Big] \tag{8}$$

The framework presumes that $p_\eta(Z)$ adheres to a Gaussian distribution, delineated in Equation (9), with Z derived through Gaussian sampling as per Equation (10). Herein, μ represents the mean, σ denotes the variance, and i is the index of the sample.

$$p_\eta(Z) \sim N(0, 1) \tag{9}$$

$$q_\beta\Big(Z \mid X^{(i)}\Big) \sim N\Big(\mu^{(i)}, \sigma^{2(i)}\Big) \tag{10}$$

The loss function of this model comprises two components: the Kullback–Leibler (KL) divergence and the reconstruction loss, with the inferable outcomes delineated in Equation (11). Here, x_i signifies the encoder network's input, and x_i' denotes the output of the decoder network.

$$\zeta = \frac{1}{2n}\sum_{j=1}^{n}\left\{\sum_{j=1}^{n}\mu_j^2 + \sigma_j^2 - 1 - \log\sigma_j^2\right\} + \frac{1}{2n}\sum_{i=1}^{n}\left\|X_i - X_i'\right\|^2 \tag{11}$$

From the foregoing equation, the term $D_{KL}\big(q_\beta(Z \mid X) \| p_\eta(Z)\big)$ represents the approximation capability of the approximate posterior model, while $\mathrm{E}_{q_\beta(Z|X^{(i)})}\Big[\log p_\eta\Big(X^{(i)} \mid Z\Big)\Big]$ signifies the reconstructive ability of the generative model to regenerate X' from Z. Consequently, this methodology can be employed to derive low-dimensional features from the initial state of crude oil storage and transportation dispatch, thereby attaining a reconstructed state that mirrors the description of the original state information to the greatest extent feasible.

Algorithm 1 Steps of computation in low-dimensional feature generation module

1: Initialize: $\mathcal{D}$, $q_\beta(Z \mid X)$, $p_\eta(X \mid Z)$, β, η
2: **while** (β, η) not convergence **do**
3: $\quad \mathcal{M} \sim \mathcal{D}$
4: $\quad Z \leftarrow$ Random sample from Gaussian distribution$\mathcal{N}(\mu, \sigma^2)$
5: $\quad$ Compute ζ and its gradients
6: $\quad$ Update (β, η)
7: **end while**
8: **return** β, η

4.3. Policy Learning Module

Leveraging the low-dimensional feature generation module, it is possible to produce a low-dimensional feature vector of the environment's original state, which facilitates the ensuing policy learning process. To guarantee the efficiency of policy training, the policy generation module in this study adopts the SAC framework as the principal structure for policy learning. This framework, predicated on the theory of entropy maximization, ensures that network updates equilibrate the maximization of expected returns with entropy,

thereby enhancing the network's exploration capabilities and expediting the learning process. The objective function is articulated in Equation (12).

$$\pi^* = \arg\max_{\pi} \mathrm{E}_{s_t,a_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t) + \alpha H(\pi(\cdot \mid s_t))\right] \tag{12}$$

$$H(\pi(\cdot \mid s_t)) = \mathrm{E}[-\log \pi(\cdot \mid s_t)] \tag{13}$$

In Equation (12), r denotes the reward function, and γ is the discount factor, while α signifies the entropy regularization coefficient, employed to modulate the significance of entropy in the learning process. In Equation (13), H represents the entropy value. A greater entropy value corresponds to a heightened level of exploration by the agent, promoting a more thorough investigation of the action space.

The training network within this framework comprises a policy network π_ϕ, an action value network $Q_{\theta_1,\theta_2}(a_t, s_t)$, and a target network, which are parameterized by Φ, θ_1, and θ_2, respectively. The action value network $Q_{\theta_1,\theta_2}(a_t, s_t)$ incorporates a dual Q-network structure. The soft Q-value is determined by taking the minimum value from two Q-value functions parameterized by θ_1 and θ_2. This approach is designed to mitigate the overestimation of inappropriate Q-values and to enhance the speed of training. The soft Q-value function is refined by minimizing the Bellman error, as detailed in Equation (15).

$$J_{Q(\theta)} = \mathrm{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\Big(Q_{\theta_{i=1,2}}(s_t,a_t) - \big(r(s_t,a_t) + \gamma V_{\bar{\varphi}}(s_{t+1})\big)\Big)^2\right] \tag{14}$$

$$V_{\bar{\varphi}}(s_{t+1}) = Q_{\bar{\theta}}(s_{t+1},a_{t+1}) - \alpha \log\big(\pi_\phi(a_{t+1} \mid s_{t+1})\big) \tag{15}$$

where $V_{\bar{\varphi}}(s_{t+1})$ represents the state value of the agent at time $t+1$, and $Q_{\bar{\theta}}(s_{t+1}, a_{t+1})$ can be estimated using the target network.

Policy network π_ϕ is updated by minimizing the KL divergence, as shown in Equation (16).

$$J_{\pi(\phi)} = \mathrm{E}_{a_t\sim\pi, s_t\sim D}\left[\log \pi_\phi(s_t,a_t) - \min_{i=1,2} Q_{\theta_i}(s_t,a_t)\right] \tag{16}$$

The proposed VSCS method is outlined in Algorithm 2.

Algorithm 2 The proposed VSCS Algorithm

Initialize: N_{encoder} in VAE, θ_1, θ_2, ϕ in Q network and policy network .
$\overline{\theta_1} = \theta_1, \quad \overline{\theta_2} = \theta_2$. Initialize experience buffer $\mathcal{D}$
for each iteration **do**
  for each environment step **do**
    $a_t = \pi_\phi(a_t|s_t)$
    $s_{t+1} = p(s_{t+1}|s_t, a_t)$
    $s'_t = N_{\text{enc}}(s_t)$
    $s'_{t+1} = N_{\text{enc}}(s_{t+1})$
    $\mathcal{D} = \mathcal{D} \cup \{s'_t, a_t, r_t, s\prime_{t+1}\}$
  end for
  for each gradient step **do**
    Sample from $\mathcal{D}$;
    Calculate the loss and update the action value network according to Equations (14) and (15)
    Calculate the loss and update the policy network according to Equation (16)
    Update the entropy regularization coefficient α
    Update the parameters of the target Q-network
  end for
end for

5. Experiment

To validate the efficacy of the proposed approach, this study conducts comprehensive experiments on the crude oil scheduling problem. The experiments include the following:

- Comparing the VSCS method introduced in this study with baseline algorithms using a dataset of refinery crude oil storage and transportation scheduling from an actual scenario.
- Analyzing the performance of the algorithm at various compression scales to determine the optimal low-dimensional feature dimensionality.
- Conducting a similarity analysis between low-dimensional reconstructed state features and original state samples and proposing a state reconstruction threshold for refinery crude oil scheduling problems based on reconstruction similarity.
- Evaluating the performance of the proposed algorithm by visualizing the low-dimensional features.

The goal of these experiments is to thoroughly assess the advantages and practical applicability of the proposed VSCS method in real-world crude oil scheduling tasks.

5.1. Data for Simulator

This investigation employs a dataset from a bona fide operational context within an oil company, encompassing various node types and their attributes, such as oil tankers, terminal tanks, commercial storage tanks, in-plant tanks, and processing devices, as delineated in Section 3. The dataset details encompass tanker oil load by type and volume, the initial liquid levels in storage tanks, the types of oil they house, storage capacities, transfer capabilities, and their processing apparatus' schemes and capacities. Integral to this study's reinforcement learning framework, the simulator accurately emulates the intricate and dynamic processes of crude oil storage and transportation within a refinery. The experimental setup utilizes a single oil tanker, 14 terminal storage tanks, 9 in-plant storage tanks, and 2 processing devices. This simulator facilitates an interactive learning milieu for the proposed algorithm, enabling adaptive training against the evolving dynamics of the refinery environment, providing continual feedback throughout the training phase, and assessing the algorithm's efficacy. The data input for the low-dimensional feature generation module is derived from sampling the experience pool within the aforementioned simulation environment, with a sampling scale consisting of 2048 random state samples, each with 61 dimensions.

In this study, the benchmark comparison is conducted against the SAC algorithm, a model premised on entropy maximization theory [38,39]. This approach ensures that updates to the training network balance the maximization of expected returns with entropy, thereby enhancing the algorithm's capacity for exploration and expediting the learning process.

5.2. Comparison with Baseline Algorithm

This section evaluates the enhanced performance of the proposed VSCS algorithm with respect to training convergence speed and the value of the final reward obtained post learning. To assess the stability of the algorithm following state reconstruction via VAE, the SAC algorithm is employed for baseline comparison. The experimental procedure involved multiple tests using diverse random seeds to determine the average learning efficacy of both the proposed algorithm and the baseline algorithm across ten different sets of random seeds. The learning performance is depicted through an average learning curve for clarity. Furthermore, in the experimental results, the rewards are logarithmically transformed for more coherent representation, as depicted in Figure 4. The principal parameters for the proposed VSCS algorithm is summarized in Table 2.

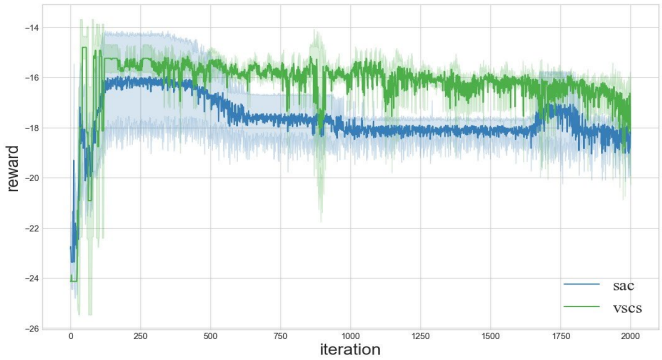

Figure 4. Learning curves of comparison methods. The solid lines show the means of 10 trials, and lighter shading shows standard errors.

Table 2. Main experimental parameters.

Model	Number of Neurons	Number of Hidden Layers	Optimizer	Discount Factor	Learning Rate	Soft Update Coefficient	Batch Size	Entropy Threshold	Experience Buffer Size
Policy learning module	512	5	Adam [40]	0.99	0.03	0.005	128	0.9	100,000
Low−dimensional feature generation module	40	1	Adam [40]						

Table 2 demonstrates that the VSCS algorithm proposed in this study markedly outperforms the baseline algorithm regarding the final reward value attained, showcasing an 89.3% enhancement in the final average reward value. In terms of training efficiency, the VSCS algorithm achieves the maximum reward in just 47 iterations. This represents a 77.5% increase in the rate at which training attains a stable state compared with the baseline algorithm. Additionally, the VSCS algorithm exhibits superior training stability relative to the baseline.

The reconstruction and compression of the state dimension prior to training the SAC network results in a significant reduction in the required sample size during the training process. This efficiency gain in sample size directly translates to enhanced network training efficiency, as the model can achieve comparable or superior learning outcomes with fewer data points.

5.3. Impact of Reconstruction with Different Compression Sizes

To assess the impact of the proposed VSCS algorithm on convergence speed and stability across varying compression scales, we conducted tests with dimensionalities set at 10, 15, 20, 25, 30, 35, 40, 45, 50, and 55. For each dimensionality, three sets of randomized trials were performed, with the average learning curves serving as the evaluative metric. The results of the learning curves are presented in Figure 5, and the algorithmic improvement rates are detailed in Table 3.

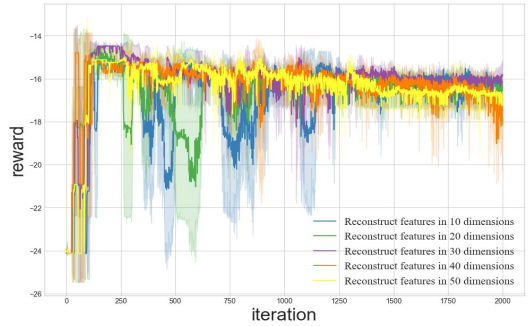

Figure 5. Comparison of low-dimensional feature reconstruction performance in different dimensions. The solid lines show the means of 3 trials, and lighter shading shows standard errors.

Table 3. Results of Comparison Methods.

	Iterations for Maximum Reward	Final Reward	Training Time to Steady State
SAC	209	−27,540,217	305
VSCS	47	−2,942,594	78
Improvement Rate (%)	77.5	89.3	74.4

Figure 5 reveals that the training process experiences increased instability when the algorithm is compressed to scales of 10 and 20, which is attributable to excessive compression that results in the loss of substantial state information. Conversely, compression scales of 30, 40, and 50 demonstrate relative stability, with the scale of 30 yielding the most effective learning strategy.

Table 4 illustrates improvements in algorithm training efficiency for the VSCS algorithm at various compression scales, with the exception of scale 15, over the baseline algorithm. Notably, at scale 40, the VSCS algorithm required only 47 rounds to achieve the cumulative maximum reward for the first time—a 77.51% increase in the rate of reaching a steady training state compared with the baseline. Furthermore, the learning performance of the VSCS algorithm was enhanced across all scales, showing an improvement rate exceeding 82%. The scales of 30 and 45 demonstrated the most significant enhancements, with an improvement rate of 92.95% in leaning performance compared with the baseline.

Table 4. The VSCS algorithm improvement rate analysis.

Feature Dimension	Iterations for Steady State	Convergence Speed Improvement Rate	Final Reward	Reward Improvement Rate
VSCS (10)	148	29.19%	−4,040,694	85.33%
VSCS (15)	215	−2.87%	−1,980,772	92.81%
VSCS (20)	158	24.40%	−2,143,448	92.22%
VSCS (25)	134	35.89%	−3,493,724	87.31%
VSCS (30)	147	29.67%	−1,940,762	92.95%
VSCS (35)	170	18.66%	−2,942,594	89.32%
VSCS (40)	47	77.51%	−2,991,348	89.14%
VSCS (45)	87	58.37%	−1,941,364	92.95%
VSCS (50)	105	49.76%	−4,876,383	82.29%

5.4. Reconstructed State Vector Similarity Analysis

In this analysis, we investigate the fidelity of state reconstruction by examining the similarity between the compressed and original states. We use the reconstruction distance to elucidate the reasons behind the enhanced training performance observed with reconstructed state vectors and introduce a threshold for reconstruction error tailored to the challenges of refinery crude oil storage and transportation scheduling. The experiment evaluates the encoder network of the VAE at compression scales of 10, 15, 20, 25, 30, 35, 40, 45, 50, and 55. We assess the congruence between 2048 original state samples and their reconstructed counterparts, which are produced by the decoder network, using Euclidean distance. The results, reflecting the similarity of output samples, are detailed in Table 5.

Table 5. Reconstruction distance analysis.

Dimensionality	55	50	45	40	35	30	25	20	15	10
Arithmetic Mean	12.66	12.72	12.66	12.61	12.67	12.47	12.52	12.55	12.53	12.54
Maximum	146.93	149.13	141.87	145.71	149.14	146.29	141.77	136.13	134.52	139.56
Minimum	0.23	0.27	0.33	0.44	0.38	0.56	0.61	1.17	1.36	0.59
Variance	608.91	620.99	617.28	608.06	619.23	606.81	611.70	614.83	614.93	611.88
Standard Deviation	24.67	24.92	24.85	24.66	24.88	24.63	24.73	24.80	24.80	24.74
Median	4.19	4.17	4.04	4.06	3.99	3.88	3.90	3.79	3.73	3.86

The encoder network with a compression scale of 30 demonstrates notable performance, yielding the highest mean similarity for reconstructed states. As detailed in Table 5, the arithmetic mean of similarity scores stands at 12.47, with a variance of 606.8.

After rigorous experimental analysis, it was determined that the reconstruction error threshold for refinery crude oil storage and transportation scheduling problems should be set at 12.47. This threshold implies that when the similarity distance falls below 12.47, the network is deemed to have achieved the standard of reconstruction.

5.5. Visual Analysis of Low-Dimensional Features

In this section, we delve into the characteristics of reconstructed states via low-dimensional visualization to elucidate the optimal effect achieved by compressing to 30 dimensions. The experiment involved reducing the dimensionality of 500 reconstructed state samples, across 10, 20, 30, 40, and 50 dimensions, down to a 2-dimensional plane using the UMAP technique [41]. We then observed the distribution of samples within this plane, employing cumulative average intracluster distance and intracluster density as metrics for quantitative analysis of the low-dimensional spatial formation. For the UMAP method, the approximate nearest-neighbor number parameter was set to 5, with the minimum interpoint distance parameter fixed at 0.3. The outcomes, displayed in Figure 6, reveal that in the two-dimensional space, the reconstructed states form clusters. Notably, the clusters at 30, 40, and 50 dimensions are more densely packed, whereas those at 10 and 20 dimensions exhibit greater dispersion.

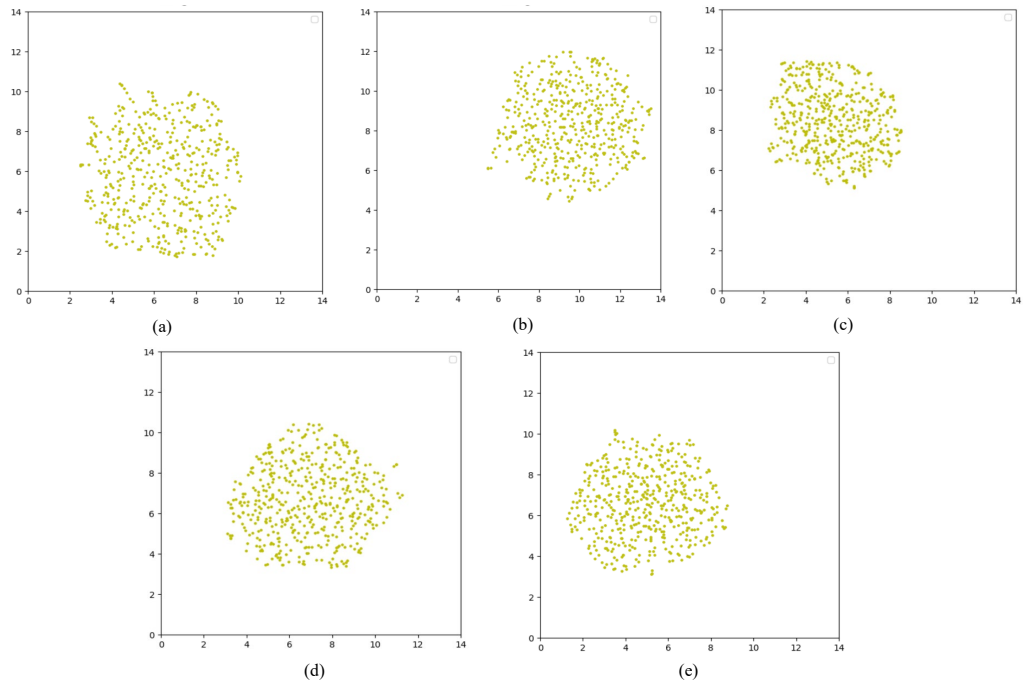

Figure 6. Results of sample dimensionality reduction visualization: (**a**) 10-dimensional sample dimensionality reduction visualization, (**b**) 20-dimensional sample dimensionality reduction visualization, (**c**) 30-dimensional sample dimensionality reduction visualization, (**d**) 40-dimensional sample dimensionality reduction visualization, (**e**) 50-dimensional sample dimensionality reduction visualization.

The quantitative analysis, utilizing the cumulative average intracluster distance (outlined in Equation (17)) and the intracluster density (specified in Equation (18)), is detailed in Table 6. Throughout the analysis, five distinct parameter configurations were employed for the assessment of means. As evidenced by Table 6, within the 30-dimensional reconstruction, the cumulative average intracluster distance is recorded at 64.10, with an average intracluster density of 0.0968—both metrics represent the most favorable values

among the five parameter sets examined. These findings indicate that the 30-dimensional reconstruction yields the most cohesive cluster structure within the sample distribution.

$$d_{\text{avgdist}} = \sum_{i \in D} \sum_{j \in D} \text{dist}(x_i, x_j) / n_D \tag{17}$$

$$d_{\text{avgcent}} = \sum_{i \in D} \text{dist}(x_i, x_{\text{center}}) / n_D \tag{18}$$

Table 6. Quantitative analysis of visualization.

	50	**40**	**30**	**20**	**10**
Intracluster Cumulative Distance (5, 0.3)	75.07	73.04	71.88	79.53	90.4
Intracluster Cumulative Distance (5, 0.15)	66.32	69.14	65.7	72.67	83.5
Intracluster Cumulative Distance (10, 0.15)	57.81	57.77	57.55	61.17	70.1
Intracluster Cumulative Distance (10, 0.10)	56.38	55.4	55.14	59.02	67.3
Intracluster Cumulative Distance (10, 0.50)	71.25	71.13	70.22	75.74	86.35
Average Intracluster Cumulative Distance	65.37	65.30	64.10	69.63	79.53
Intracluster Density (10, 0.50)	0.104	0.104	0.102	0.109	0.124
Intracluster Density (10, 0.15)	0.083	0.082	0.082	0.086	0.101
Intracluster Density (5, 0.3)	0.108	0.104	0.104	0.113	0.131
Intracluster Density (5, 0.15)	0.094	0.099	0.093	0.105	0.124
Average Intracluster Density	0.0984	0.0984	0.0968	0.1042	0.1208

6. Conclusions

This study introduces the VSCS algorithm to expedite the training process of deep reinforcement learning models. The VSCS framework incorporates two key components: a low-dimensional feature generation module and a policy learning module. The former serves as a pretraining phase, leveraging a VAE to faithfully encapsulate the original state information within a reduced feature space. Upon completion of the training, the low-dimensional feature generation module integrates into the primary framework, furnishing the policy learning module with compact feature representations for policy network training. This synergistic approach facilitates end-to-end learning across both modules. A novel methodology was also developed to ascertain the optimal dimensionality for these low-dimensional features, accounting for reconstruction fidelity and visual analysis outcomes. A comprehensive experiment with the proposed method on the crude oil scheduling problem not only confirmed the efficacy of the framework but also empirically validated the optimal selection of low-dimensional feature dimensions.

The methodology presented herein primarily addresses the enhancement of performance in deep reinforcement learning when confronted with large-scale state representations. While it has yielded promising results, the prospect of its application within the industrial sector necessitates additional thorough investigation. Future research directives could include conducting generalizability studies on scheduling decisions across various refineries to solidify the method's applicability and robustness in diverse industrial contexts.

Author Contributions: N.M., conceptualization, methodology, software, formal analysis, visualization, writing—original draft, and writing—review and editing; H.L. (Hongqi Li), supervision, writing—original draft, and writing—review and editing; H.L. (Hualin Liu), formal analysis, writing—original draft, and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data are not publicly available due to restrictions, their containing information that could compromise the privacy and interest of company.

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

References

1. Xu, J.; Zhang, S.; Zhang, J.; Wang, S.; Xu, Q. Simultaneous scheduling of front-end crude transfer and refinery processing. *Comput. Chem. Eng.* **2017**, *96*, 212–236. [CrossRef]
2. Jia, Z.; Ierapetritou, M.; Kelly, J.D. Refinery short-term scheduling using continuous time formulation: Crude-oil operations. *Ind. Eng. Chem. Res.* **2003**, *42*, 3085–3097. [CrossRef]
3. Zheng, W.; Gao, X.; Zhu, G.; Zuo, X. Research progress on crude oil operation optimization. *CIESC J.* **2021**, *72*, 5481.
4. Hamisu, A.A.; Kabantiok, S.; Wang, M. An Improved MILP model for scheduling crude oil unloading, storage and processing. In *Computer Aided Chemical Engineering*; Elsevier: Lappeenranta, Finland, 2013; Volume 32, pp. 631–636.
5. Zhang, H.; Liang, Y.; Liao, Q.; Gao, J.; Yan, X.; Zhang, W. Mixed-time mixed-integer linear programming for optimal detailed scheduling of a crude oil port depot. *Chem. Eng. Res. Des.* **2018**, *137*, 434–451. [CrossRef]
6. Furman, K.C.; Jia, Z.; Ierapetritou, M.G. A robust event-based continuous time formulation for tank transfer scheduling. *Ind. Eng. Chem. Res.* **2007**, *46*, 9126–9136. [CrossRef]
7. Li, F.; Qian, F.; Du, W.; Yang, M.; Long, J.; Mahalec, V. Refinery production planning optimization under crude oil quality uncertainty. *Comput. Chem. Eng.* **2021**, *151*, 107361. [CrossRef]
8. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef] [PubMed]
9. Esteso, A.; Peidro, D.; Mula, J.; Díaz-Madroñero, M. Reinforcement learning applied to production planning and control. *Int. J. Prod. Res.* **2023**, *61*, 5772–5789. [CrossRef]
10. Dong, Y.; Zhang, H.; Wang, C.; Zhou, X. Soft actor-critic DRL algorithm for interval optimal dispatch of integrated energy systems with uncertainty in demand response and renewable energy. *Eng. Appl. Artif. Intell.* **2024**, *127*, 107230. [CrossRef]
11. Kuhnle, A.; Kaiser, J.P.; Theiß, F.; Stricker, N.; Lanza, G. Designing an adaptive production control system using reinforcement learning. *J. Intell. Manuf.* **2021**, *32*, 855–876. [CrossRef]
12. 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. [CrossRef]
13. Yang, X.; Wang, Z.; Zhang, H.; Ma, N.; Yang, N.; Liu, H.; Zhang, H.; Yang, L. A review: Machine learning for combinatorial optimization problems in energy areas. *Algorithms* **2022**, *15*, 205. [CrossRef]
14. Ogunfowora, O.; Najjaran, H. Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. *J. Manuf. Syst.* **2023**, *70*, 244–263. [CrossRef]
15. Hamisu, A.A.; Kabantiok, S.; Wang, M. Refinery scheduling of crude oil unloading with tank inventory management. *Comput. Chem. Eng.* **2013**, *55*, 134–147. [CrossRef]
16. Shah, N. Mathematical programming techniques for crude oil scheduling. *Comput. Chem. Eng.* **1996**, *20*, S1227–S1232. [CrossRef]
17. Pinto, J.M.; Joly, M.; Moro, L.F.L. Planning and scheduling models for refinery operations. *Comput. Chem. Eng.* **2000**, *24*, 2259–2276. [CrossRef]
18. Zimberg, B.; Ferreira, E.; Camponogara, E. A continuous-time formulation for scheduling crude oil operations in a terminal with a refinery pipeline. *Comput. Chem. Eng.* **2023**, *178*, 108354. [CrossRef]
19. Su, L.; Bernal, D.E.; Grossmann, I.E.; Tang, L. Modeling for integrated refinery planning with crude-oil scheduling. *Chem. Eng. Res. Des.* **2023**, *192*, 141–157. [CrossRef]
20. Castro, P.M.; Grossmann, I.E. Global optimal scheduling of crude oil blending operations with RTN continuous-time and multiparametric disaggregation. *Ind. Eng. Chem. Res.* **2014**, *53*, 15127–15145. [CrossRef]
21. Assis, L.S.; Camponogara, E.; Menezes, B.C.; Grossmann, I.E. An MINLP formulation for integrating the operational management of crude oil supply. *Comput. Chem. Eng.* **2019**, *123*, 110–125. [CrossRef]
22. Assis, L.S.; Camponogara, E.; Grossmann, I.E. A MILP-based clustering strategy for integrating the operational management of crude oil supply. *Comput. Chem. Eng.* **2021**, *145*, 107161. [CrossRef]
23. Zimberg, B.; Camponogara, E.; Ferreira, E. Reception, mixture, and transfer in a crude oil terminal. *Comput. Chem. Eng.* **2015**, *82*, 293–302. [CrossRef]
24. Ramteke, M.; Srinivasan, R. Large-scale refinery crude oil scheduling by integrating graph representation and genetic algorithm. *Ind. Eng. Chem. Res.* **2012**, *51*, 5256–5272. [CrossRef]
25. Hou, Y.; Wu, N.; Zhou, M.; Li, Z. Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *47*, 517–530. [CrossRef]
26. Hou, Y.; Wu, N.; Li, Z.; Zhang, Y.; Qu, T.; Zhu, Q. Many-objective optimization for scheduling of crude oil operations based on NSGA-III with consideration of energy efficiency. *Swarm Evol. Comput.* **2020**, *57*, 100714. [CrossRef]
27. Ramteke, M.; Srinivasan, R. Integrating graph-based representation and genetic algorithm for large-scale optimization: Refinery crude oil scheduling. In *Computer Aided Chemical Engineering*; Elsevier: Amsterdam, The Netherlands, 2011; Volume 29, pp. 567–571.
28. Badia, A.P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, Z.D.; Blundell, C. Agent57: Outperforming the atari human benchmark. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 13–18 July 2020; pp. 507–517.
29. Hubbs, C.D.; Li, C.; Sahinidis, N.V.; Grossmann, I.E.; Wassick, J.M. A deep reinforcement learning approach for chemical production scheduling. *Comput. Chem. Eng.* **2020**, *141*, 106982. [CrossRef]

30. Gui, Y.; Tang, D.; Zhu, H.; Zhang, Y.; Zhang, Z. Dynamic scheduling for flexible job shop using a deep reinforcement learning approach. *Comput. Ind. Eng.* **2023**, *180*, 109255. [CrossRef]
31. Che, G.; Zhang, Y.; Tang, L.; Zhao, S. A deep reinforcement learning based multi-objective optimization for the scheduling of oxygen production system in integrated iron and steel plants. *Appl. Energy* **2023**, *345*, 121332. [CrossRef]
32. Lee, Y.H.; Lee, S. Deep reinforcement learning based scheduling within production plan in semiconductor fabrication. *Expert Syst. Appl.* **2022**, *191*, 116222. [CrossRef]
33. Yang, F.; Yang, Y.; Ni, S.; Liu, S.; Xu, C.; Chen, D.; Zhang, Q. Single-track railway scheduling with a novel gridworld model and scalable deep reinforcement learning. *Transp. Res. Part Emerg. Technol.* **2023**, *154*, 104237. [CrossRef]
34. Pan, L.; Cai, Q.; Fang, Z.; Tang, P.; Huang, L. A deep reinforcement learning framework for rebalancing dockless bike sharing systems. In Proceedings of the AAAI Conference on Artificial Intelligence, Hilton, HI, USA, 27 January–1 February 2019; Volume 33, pp. 1393–1400.
35. Yan, Q.; Wang, H.; Wu, F. Digital twin-enabled dynamic scheduling with preventive maintenance using a double-layer Q-learning algorithm. *Comput. Oper. Res.* **2022**, *144*, 105823. [CrossRef]
36. Chen, Y.; Liu, Y.; Xiahou, T. A deep reinforcement learning approach to dynamic loading strategy of repairable multistate systems. *IEEE Trans. Reliab.* **2021**, *71*, 484–499. [CrossRef]
37. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
38. Zang, W.; Song, D. Energy-saving profile optimization for underwater glider sampling: The soft actor critic method. *Measurement* **2023**, *217*, 113008. [CrossRef]
39. Hussain, A.; Bui, V.H.; Musilek, P. Local demand management of charging stations using vehicle-to-vehicle service: A welfare maximization-based soft actor-critic model. *eTransportation* **2023**, *18*, 100280. [CrossRef]
40. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
41. McInnes, L.; Healy, J.; Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv* **2018**, arXiv:1802.03426.