國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management

College of Management

National Taiwan University

Master's Thesis

使用穩健最佳化作為深度強化學習指引 – 以化學物料生產排程作為應用案例

Robust Optimization as an Oracle Guiding Method for Deep Reinforcement Learning – an Application in Chemical Material Production Scheduling Problem

黃奕滔

Yi-Tao Huang

指導教授：李家岩 博士

Advisor: Chia-Yen Lee, Ph.D.

中華民國 112 年 7 月

July 2023

doi:10.6342/NTU202401914

國立臺灣大學碩士學位論文
口試委員會審定書

使用穩健最佳化作為深度強化學習指引—以化學物料生產排程作為應用案例

Robust Optimization as an Oracle Guiding Method for Deep Reinforcement Learning – an Application in Chemical Material Production Scheduling Problem

本論文係黃奕滔君（學號 R10725026）在國立臺灣大學資訊管理學系、所完成之碩士學位論文，於民國 112 年 7 月 19 日承下列考試委員審查通過及口試及格，特此證明

口試委員：[illegible]

[illegible] [illegible]

[illegible]

所　　長：[illegible]

doi:10.6342/NTU202301914

謝辭

首先非常感謝李家岩老師在我大四時以及碩士班兩年的指導，不管是在研究方向的指導或支持都給了我很大的幫助，原先覺得強化學習與最佳化方法的結合會太過困難，但老師在一路以來不僅沒有否定這條路線，還給予了我一些建議，讓我建立了很大的信心。除此之外，不能不感謝孔令傑老師在大學四年對我的影響，讓我有抽象化的思考能力、扎實的作業研究基礎、還有大型程式撰寫的經驗，少了老師們的培養我是絕對寫不出這篇論文的。

同時，要感謝佑鑫學長不厭其煩的和我討論強化學習的數學推導與演算法的設計，還有芯瑜在最後一步時協助我編修論文並提供先前的經驗給我參考，以及柏儒在我統計相關知識不足時給予的協助。當然，瑞昕、采嬪和彬祺給我的陪伴與鼓勵也是不可或缺的，真的很開心能夠一起走過這段旅途。

最後，我要感謝父母、女友無條件的各種支援與諒解，讓我沒有後顧之憂的進行研究，也才能順利把這篇論文產出。

doi:10.6342/NTU202401914

摘要

在本文中，我們提出了一種新的強化學習訓練框架，專門用於動態排程單階段多產品化學反應器。該方法透過最佳化方法的合作，將策略梯度收斂時所遇到的局部最優挑戰減輕，優化了強化學習模型目標值以及它的穩定性。除此之外，我們進一步透過將穩健最佳化作為指導引擎，提高模型對參數誤差的穩健性，促進了在需求與生產效率不穩定的情況下所能採用的較保守之決策風格。實驗結果證明了我們所提出方法的有效性，與簡單的演員-評論家方法相比，在同樣的參數設計下多個指標上都有所提高。

本研究的研究貢獻在深度強化學習與最佳化方法上的整合有所突破，也透過本框架提供新的強化學習訓練框架，可以應用於其他領域問題之中。

關鍵詞：製造排程、強化學習、穩健最佳化、化工製造、隨機規劃

doi:10.6342/NTU202401914

Abstract

In this paper, we present a novel guiding framework for the training phase of Reinforcement Learning (RL) models, specifically tailored for dynamic scheduling of a single-stage multi-product chemical reactor. Our approach addresses the challenge of local optima in policy gradient methods by integrating optimization methods, enhancing both the objective value and stability of the RL model. We further enhance the robustness of our model against parameter distortions by incorporating robust optimization as a guiding engine, promoting a more conservative decision-making style. Our experimental results demonstrate the efficacy of our approach, with several metrics compared to simple Actor-Critic methods.

Our work thus serves as an advancement in the integration of Deep Reinforcement Learning and optimization methods, hopefully opening new avenues for research and application in dynamic scheduling and beyond.

Keywords: Production scheduling, reinforcement learning, robust optimization, chemical production, stochastic programming

doi:10.6342/NTU202401914

Table of Contents

doi:10.6342/NTU202401914

doi:10.6342/NTU202401914

List of figures

doi:10.6342/NTU202401914

List of tables

doi:10.6342/NTU202401914

Chapter 1 Introduction

Parts of this thesis have been developed into a journal article, which has been published as *'Robust-optimization-guiding deep reinforcement learning for chemical material production scheduling'* in *Computers & Chemical Engineering* (Lee et al., 2024).

1.1 Background and Motivation

Industrial production plays a pivotal role worldwide, leading to unprecedented demand for essential chemical materials. The shortage of such materials can result in over-queued time and devastate work-in-progress. Moreover, in the era of COVID-19 and the subsequent post-COVID-19 period, supply chains have become increasingly unpredictable and variable in demand (Hoek, 2020). Chemical material production, as a crucial upstream sector for most production industries, faces significant uncertainties like production delays, yield rate fluctuation, shifting demand, and the need for frequent rescheduling (Gupta & Maravelias, 2016; Janak et al., 2007; Li & Ierapetritou, 2008).

In chemical material production planning scenarios, stockout (i.e., demand unfulfillment) situations are mostly caused by demand and yielding rate uncertainty (Gupta & Maravelias, 2016). Besides, the chemical material production station is usually non-stoppable, which means the production process is continuous. In other words, we cannot set the machine to be idle or discretely change the producing product;

doi:10.6342/NTU202401914

the product transition processes are involved with different settings, including temperature, pressure, or concentrations, etc., and the change of configurations are continuous. The switching phases will yield off-grade material that cannot be sold and are thus considered as costs (Hubbs et al., 2020). By the aforementioned characteristics, the scheduling problem becomes to determine when to change the producing product and what product to produce in the next.

The production scheduling problem is typically NP-hard, suggesting there's no existing algorithm that can solve such a problem within polynomial time (Lenstra et al., 1977). Traditional operations research methodologies, such as mixed-integer linear programming, use a branch-and-bound algorithm for finding (sub)optimal solutions. When factoring in uncertainty, stochastic programming incorporates expected values and distribution assumptions to align with real-world scenarios. However, the time-consuming nature of the optimization process poses challenges when dealing with cases requiring frequent rescheduling (Sahinidis, 2004).

Recently, Deep Reinforcement Learning (DRL) has been introduced as a potential solution to this issue (Hubbs et al., 2020). While DRL requires considerable offline training time, it offers relatively low-cost online planning (inferencing). However, the convergence of neural networks can be extremely slow if the solution space is too large (Sutton & Barto, 1998).

doi:10.6342/NTU202401914

Moreover, the dimensional complexity of discrete optimization problems often confounds the learning algorithm during exploration and exploitation phases without the aid of an "oracle guiding" mechanism or something similar. For instance, to achieve better objective value, an agent might have to sacrifice short-term rewards for long-term gains. Even though such a decision theoretically can be modeled by a multi-layer neural network, it's almost impossible for a random trial-and-error exploration method (like epsilon-greedy) to discover and learn accordingly (Silver et al., 2016). Training a decision-making style, such as robustness to randomness, becomes even more challenging.

Given the considerations and challenges in chemical production scheduling, there have been recent attempts to solve it using DRL. Nevertheless, DRL faces its own set of obstacles, including time-consuming training processes and difficulties in handling uncertainties. Therefore, this study proposes a Reinforcement Learning embedded with Robust Optimization (RLeRO) framework. It is designed to aid the DRL model in solving the scheduling problem by incorporating an "oracle guiding" mechanism in the training phase. Furthermore, this framework integrates uncertainty robustness into its decision-making style to better align with the characteristics of modern supply chains.

doi:10.6342/NTU202401914

1.2 Research Objective

The primary research objective of this study is to develop a Deep Reinforcement Learning (DRL) agent that is capable of tackling the chemical material production scheduling problem while robustly handling uncertainties.

From a detailed analysis of the chemical material production industry and the associated planning problems, we recognize that an effective scheduling methodology for this real-world application should exhibit three key attributes: (1) robustness to variations in demand and yield rate, without the need for distribution assumptions, (2) low computational burden during the inferencing of scheduling suggestions, and (3) the presence of guidance during the learning phase.

Our approach involves refining the framework proposed by Hubbs et al. (2020) that uses DRL to solve the scheduling problem in the chemical material production industry. This refined framework aims to meet all three requirements simultaneously. It will be a combination of robust optimization and an advanced actor-critic reinforcement learning approach during the training phase. In addition, we will evaluate the performance of this new framework by comparing it with traditional optimization approaches and the standard Advantage Actor-Critic (A2C) method.

The main contribution of our work is to introduce an operations-research-based guiding method in the RL training phase, which can help the policy network escape

doi:10.6342/NTU202401914

from local optimality. Furthermore, we adopted robust optimization as a guiding engine for the policy network to learn a conservative decision-making style and evaluate the value of it through several metrics.

1.3 Thesis Architecture

In Chapter 2, we will conduct a review of previous methodologies applied to chemical production scheduling, as well as oracle guiding methods. Chapter 3 introduces the main contribution of this study, the RLeRO algorithm. In Chapter 4, we will evaluate the proposed algorithm, comparing its performance with traditional optimization methods and the simple A2C as benchmarks. Finally, in Chapter 5, we will draw conclusions from our study and look forward to potential improvements and future research directions.

doi:10.6342/NTU202401914

Chapter 2 Literature Review

In this chapter, we will present previous works related to our study, respectively. Firstly, Section 2.1 surveys papers about scheduling methodologies with uncertainties. Section 2.2 reviews the applications of optimization methodologies and reinforcement learning in chemical production scheduling problem. As for Section 2.3, oracle guiding methods for DRL will be discussed.

2.1 Scheduling with Uncertainties

Several methodologies have been proposed to manage uncertainties in scheduling problems. Janak et al. (2006) present a reactive scheduling framework, which strategically fixes binary variables from the original production schedule to circumvent the need for comprehensive rescheduling. Mihoubi et al. (2021) proposes a surrogate-assisted simulation-optimization approach, based on scheduling rules, to address Reactive Scheduling (RS) and the Flexible Job Shop Scheduling Problem (FJSSP), crucial aspects of real-world manufacturing systems. It aims to encapsulate the dynamic nature of FJSSP while balancing reactivity and overall system performance.

Taking a different route, Bonfill et al. (2004) leverage a two-stage stochastic optimization approach to manage risk in short-term scheduling of multiproduct batch plants dealing with demand uncertainties. This body of work was later expanded in next year (Bonfill et al., 2005) to accommodate variable processing times in chemical batch

doi:10.6342/NTU202401914

processes' short-term scheduling. Hu et al. (2020) presents a two-stage stochastic programming framework for optimizing production in a manufacturing plant's kitting facility, accounting for uncertainties in kit demand and worker yield. Given a multi-product case study, the model effectively handles uncertainties, underlining their significant impact on production planning decisions.

In an effort to handle scheduling challenges arising from uncertain processing times, market demands, or prices, Lin et al. (2004) proposed a robust optimization method. Janak et al. (2007) built on this work, tailoring the approach for scenarios where uncertainty is described by a known probability distribution.

On the other hand, Petrovic & Duenas (2006) utilized fuzzy programming to handle parallel machine scheduling and rescheduling in an uncertain environment. Their proposed method, a predictive-reactive approach, involves two key steps: first, the creation of a schedule, and second, a rescheduling phase. Each step addresses distinct aspects of the scheduling problem.

Jia & Ierapetritou (2006a, 2006b) proposed a unique method for uncertainty analysis on the right-hand side (RHS) for mixed-integer linear programming (MILP) problems. The procedure involves an iterative process that includes sensitivity analysis using linear programming and multi-parametric linear programming, as well as updating the branch-and-bound tree. They further improved this framework by devising

doi:10.6342/NTU202401914

a way to manage the issue of infeasibility. This involved providing a description of the feasible region prior to implementing the parametric MILP algorithm. Additionally, they took into account uncertainty in the objective function coefficients and problem constraints, offering a comprehensive approach to managing uncertainties in scheduling.

Chang et al. (2022) applies deep reinforcement learning (DRL) to the dynamic and complex task of scheduling in a smart factory's production process. By designing a double deep Q-networks (DDQN) architecture and a soft ε-greedy behavior policy, it provides an approach to the flexible job shop scheduling problem (FJSP) that excels in real-time adaptation and minimizes penalties for earliness and tardiness.

To address uncertainty, two major methodologies for planning and scheduling are robust optimization (RO) and stochastic programming (Grossmann et al., 2016). In contrast to stochastic optimization, which starts by assuming the uncertainty has a probabilistic description, RO constructs a solution that is feasible for any realization of the uncertainty in a given set (Bertsimas et al., 2011). The characteristic of RO allows user to construct model without knowing underlying stochastic distribution of uncertainties, which is usually unavailable in real world problems (Nemirovski, 2019).

doi:10.6342/NTU202401914

2.2 Chemical Production Scheduling

Jung et al. (2004) proposed a multi-stage recourse model that takes demand uncertainty into consideration, which assumed the demand distribution as discrete scenarios and conduct multi-steps of wait-and-see process. However, the multi-stage recourse framework will cause exponential growth in problem size when the stage or scenario count increased. While Sand & Engell (2004) used two-stage recourse model considering both demand and yielding rate uncertainties as discrete scenarios as well.

As for RO approach, Lin et al. (2004) uses ellipsoidal set to take care demand and yielding rate uncertainties simultaneously, moreover, the robust counterpart model does not give more computation burden than the original one, which is different from multi-stage models.

The works of RL application on scheduling problem is relatively sparse but having good results, Riedmiller & Riedmiller (1999) adopted perceptron into Q-learning for production scheduling problem without considering uncertainties, which also supported the idea of our research motivation: "RL is better for frequent replanning cases because of its low computational burden in inferences." The latest paper applied RL in chemical production scheduling problem we can find is the framework from (Hubbs et al., 2020), which is the one we targeted to enhance; the paper uses A2C method to describe the unknown demand distribution without giving assumptions, but as the conclusion it

doi:10.6342/NTU202401914

mentioned: “Future research can explore possibilities for integrating DRL and optimization methods.”

Our study is to improve the RL application in chemical production scheduling problem through operations research guiding methods, including deterministic and robust optimization; noted that different guiding methods bring different decision-making style up, for instance, our study shows that RO guiding in the training phase will encourage network to take robust decisions.

The comparisons of previous works in chemical material production scheduling problems with uncertainty are shown in Table 1.

doi:10.6342/NTU202401914

Table 1 Comparisons of previous works in chemical material production scheduling problems with uncertainty

Paper	Methodology	Uncertainty		Decision making style	Distribution assumption	Computational time
		Demand	Yielding rate			
Jung et al. (2004)	Multi-stage recourse	O	X	Wait-and-see, conservative before uncertainty revealed	Required, separated into discrete scenarios	Not considered
Sand & Engell (2004)	Two-stage recourse	O	O	Wait-and-see, conservative before uncertainty revealed	Required, separated into discrete scenarios	Not considered
Lin et al. (2004)	Robust optimization	O	O	Robust to uncertainty at the beginning	Not required	Not considered
Riedmiller & Riedmiller (1999)	Q-learning	X	X	Target to maximum returns	Not required	Considered
Hubbs et al. (2020)	A2C	O	X	Target to maximum returns	Not required	Considered
This study	A2C + OR guiding	O	O	Self-definable (Robust as an example)	Not required	Considered

doi:10.6342/NTU202401914

2.3 Oracle Guiding DRL

Although there are several algorithms helps the convergence of neural network (NN) during the training phase, “an important but under-explored aspect is how to leverage oracle observation to facilitate learning.” (Han et al., 2022). i.e., information that is invisible during online decision making, but is available during offline training is usually unused. They combines Bayesian probability concept with oracle provided information, proposed a VLOG algorithm that significantly reduce training time in each RL classical cases.

Suphx, an agent of playing a traditional Chinese game Mahjong developed by Microsoft, provides perfect information while training and gradually decay the information completeness from 1 to 0, which is sort of implementing a training guidance that instruct the learning process get into the swing faster (Li et al., 2020).

Another state-of-the-art DRL project is AlphaGo developed by Google, which also adopt transfer learning as an approach of guidance instead of trial-and-error while the neuron network is ignorant at first. The information is not explicitly exposed to the learning algorithm but implicitly embedded in the pretrained model. The pretrained model took the games performed by human professional chess player as training set, and it learnt to predict next move after observing current status (Silver et al., 2016). It is a little different from VLOG and Suphx cases but with similar concept.

doi:10.6342/NTU202401914

From the literatures surveyed so far, researches in the guidance of reinforcement learning are relatively lack. Although the robustness of RL algorithm is widely discussed, most of the literatures uses adversarial approach for training, which can hurt the generalization of model (Raghunathan et al., 2019) and with the disadvantage of non-convergence, overfitting, and mode-collapse (Andriushchenko & Flammarion, 2020). Therefore, our research proposed a “Reinforcement Learning embedded with Robust Optimization” training framework; targeting to achieve the robustness in parameters and observations perturbations and boost the effectiveness of training progress.

Table 2 shows the comparisons of previous works which adopted different kinds of guidance in the RL training phase.

Table 2 Comparisons of previous works adopted guidance in RL training phase

Paper	Training phase guiding	Target
(Han et al., 2022)	Bayesian probability	Shorten training phase Improve explainability
(J. Li et al., 2020)	Decaying perfect information	Shorten training phase Help convergence
(Silver et al., 2016)	Network transfer learning	Shorten training phase Improve policy

doi:10.6342/NTU202401914

This research	Operations research	Build up decision style

doi:10.6342/NTU202401914

Chapter 3 Methodologies

The research framework is demonstrated in Section 3.1. In Section 3.2 we will describe the chemical material production scheduling problem as a mathematical model for a more formal and unambiguity illustration. Sequentially, the definition and details of the reinforcement learning model will be explained in Section 3.3. Lastly, in Section 3.4, the OR-guiding algorithm is proposed.

3.1 Research Framework

The research framework is shown in Figure 1 Research framework:

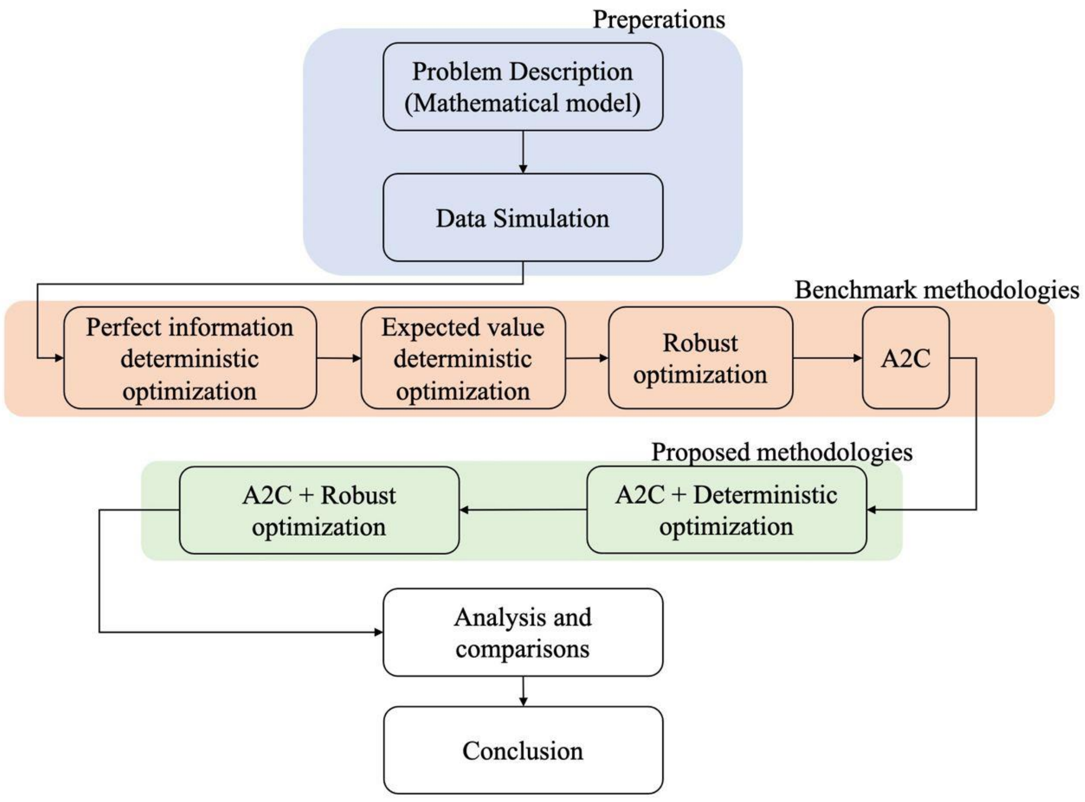

Preperations
Problem Description (Mathematical model)
Data Simulation
Benchmark methodologies
Perfect information deterministic optimization
Expected value deterministic optimization
Robust optimization
A2C
Proposed methodologies
A2C + Deterministic optimization
A2C + Robust optimization
Analysis and comparisons
Conclusion

Figure 1 Research framework

In the preparatory stage, we first construct a mathematical programming model for the chemical material production scheduling problem, designing the configurations of

doi:10.6342/NTU202401914

parameters accordingly.Next, we implement methodologies including deterministic optimization with perfect information and expected value, robust optimization, and Advantage Actor Critic (A2C) as benchmarks.The third part introduces the operations research-guiding (OR-guiding) methodologies we propose, transitioning from A2C with deterministic optimization guidance to A2C with robust optimization guidance. Lastly, we carry out numerical analyses and result comparisons to draw conclusions for each model and provide suggestions for the use case of each methodology.

3.2 Problem Definition

3.2.1 Problem Description

The problem definition of this study is based on the mathematical model constructed by Hubbs et al. (2020) with slight modification; we introduce the yielding rate uncertainty into the parameters, which is not included in the baseline paper.

We defined an experiment period set $\hat{P}$ (each period is one day), which contains all of the periods in a whole experiment, and a planning period set P rolls in $\hat{P}$ for each scheduling subproblems. For each time we plan or replan a schedule is to solve such a subproblem with period set P. Inside of the planning set P, we can distinguish two types of periods: fixed planning periods set F and the remaining are unfixed planning periods, where F is the first $|F|$ periods of P. The implication of fixed periods is that once a planning schedule has been determined, it cannot be changed due

doi:10.6342/NTU202401914

to the operating procedures defined by the chemical plant. This implies that the main decision we make in a schedule pertains to the last $|P| - |F|$ periods. The main idea of experiment period, planning period, and fixed period in Figure 2 Example of experiment setting are provided as an example, with the setting of $|\hat{P}| = 30$ |, $|\mathrm{P}| = 12$, and $|F| = 5$.

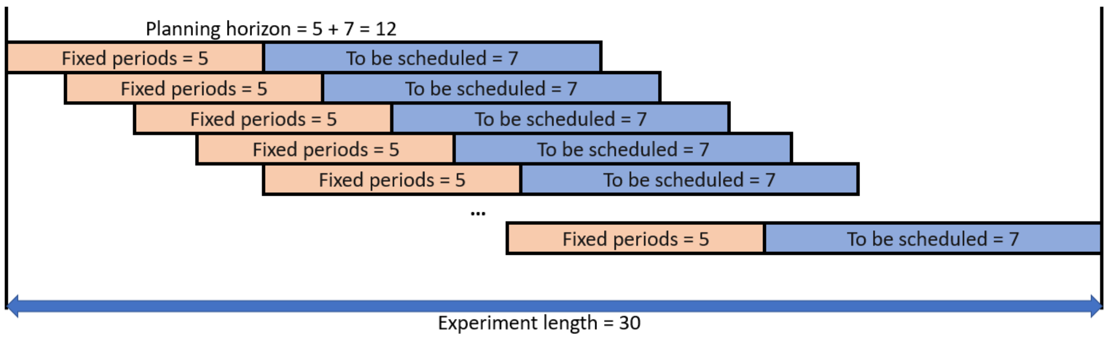


Figure 2 Example of experiment setting

In the scheduling problem, we must determine which type of chemical material product from the product set I the machine should produce. The costs that should be taken into consideration include the transition cost C_{ij}^{T}, which is incurred when transitioning from product $i \in I$ to $j \in I$; the unit inventory cost C_i^S for product $i \in I$; and the stockout cost C_i^L for product $i \in I$.

The reason for the transition cost is that switching products often leads to large fluctuations in processing temperature, which in turn yields off-grade material that cannot be sold and is thus counted as a discount term (Hubbs et al., 2020). Furthermore, the unit profit V_i represents the reward for fulfilling the demand for each product $i \in$

doi:10.6342/NTU202401914

I. The decision-making process aims to maximize the gross profit, i.e., the sum of the profits V_i minus the costs C_{ij}^T, C_i^S, and C_i^L.

It is crucial to note that the demand $\widetilde{D}_{ip}$ and yielding rate $\tilde{A}_i$ are uncertain, meaning we cannot know the actual values (noted as $\widehat{D}_{ip}$ and $\hat{A}_{ip}$) until the uncertainty is revealed over time. However, the value of $\widetilde{D}_{ip}$ will gradually become more accurate as the current period gets closer to the predicted demand period in our experiment design, the details of which will be provided in the data simulation section.

3.2.2 Mathematical Model

As the scheduling problem described above, the sets and parameters are listed below:

Sets

I	Product categories set.
$\hat{P}$	Experiment periods set.
P	Planning periods set, which rolls in $\hat{P}$ for each scheduling subproblems.
F	Fixed planning periods set, which is a subset of P (the first $\lvert F \rvert$ periods of P).

Parameters

C_{ij}^T	Transition cost from producing product i to j, $\forall i \in I, j \in I$.
C_i^S	Unit inventory cost of product i, $\forall i \in I$.
C_i^L	Unit stockout cost of product i, $\forall i \in I$.
V_i	Unit profit of product i, $\forall i \in I$.

doi:10.6342/NTU202401914

S_i^I Initial inventory of product i, $\forall i \in I$.

X_{if} Fixed production schedule for product i in period f, $\forall i \in I, f \in F$.

$\widetilde{D}_{ip}$ Predicted uncertain demand for product i in period p, $\forall i \in I, p \in P$.

$\tilde{A}_i$ Nominal uncertain yield amount for product i in unit period, $\forall i \in I$.

$\widehat{D}_{ip}$ Actual demand amount for product i in period p, which is not available when planning, $\forall i \in I, p \in P$.

$\hat{A}_{ip}$ Actual production amount for product i in period p, which is not available when planning, $\forall i \in I, p \in P$.

The decision variables used for the model are as follows:

Decision Variables

x_{ip} The binary variable describes whether product is scheduled in a period, 1 if product i is scheduled in period p, else 0, $\forall i \in I, p \in P \cup \{P_{-1}\}$.

Dependent Variables

y_{ip} The sales amount of product i in period p, $\forall i \in I, p \in P$.

s_{ip} The inventory amount of product i in period p, $\forall i \in I, p \in P \cup \{P_{-1}\}$.

l_{ip} Stockout amount of product i in period p, $\forall i \in I, p \in P$.

z_{ijp} The binary variable describes whether transition happened in a period, 1 if transition from producing product i to j occurred, else 0, $\forall i \in I, j \in I$.

doi:10.6342/NTU202401914

The aim of the scheduling model is to maximize the gross profit of the chemical production schedule. This includes sales profit derived from demand fulfillment, denoted as $\sum_{i\in I}\sum_{p\in P} V_i y_{ip}$, the inventory cost for product storage represented by $\sum_{i\in I}\sum_{p\in P} C_i^S s_{ip}$, the stockout cost due to unfulfilled demand indicated as $\sum_{i\in I} C_i^L l_{ip}$, and the transition cost, given by $\sum_{i\in I}\sum_{j\in I, j\neq i}\sum_{p\in P} C_{ij}^T z_{ijp}$. The objective function is illustrated in equation (1).

$$Max \sum_{i\in I}\sum_{p\in P} V_i y_{ip} - \sum_{i\in I}\sum_{p\in P} C_i^S s_{ip} - \sum_{i\in I} C_i^L l_{ip} - \sum_{i\in I}\sum_{j\in I, j\neq i}\sum_{p\in P} C_{ij}^T z_{ijp} \qquad (3.1)$$

The constraints of the model are listed below:

$$s_{iP_0} = S_i^I, \forall i \in I \qquad (3.2)$$

$$x_{if} = X_{ij}, \forall i \in I, f \in F \cup \{P_0\} \qquad (3.3)$$

$$s_{ip} = s_{i(p-1)} + \tilde{A}_i x_{ip} - \widetilde{D}_{ip} + l_{ip}, \forall i \in I, p \in P \qquad (3.4)$$

$$\sum_{i\in I} z_{ijp} = x_{jp}, \forall j \in I, p \in P \qquad (3.5)$$

$$\sum_{j\in I} z_{ijp} = x_{i(p-1)}, \forall i \in I, p \in P \qquad (3.6)$$

$$\sum_{i\in I} x_{ip} = 1, \forall p \in P \qquad (3.7)$$

$$y_{ip} = min\left(\tilde{A}_i x_{ip} + s_{i(p-1)}, \widetilde{D}_{ip}\right), \forall i \in I, p \in P \qquad (3.8)$$

$$x_{ip} \in \{0,1\}, \forall i \in I, p \in P \cup \{P_0\} \qquad (3.9)$$

$$z_{ijp} \in \{0,1\}, \forall i \in I, j \in I, p \in P \qquad (3.10)$$

$$s_{ip} \geq 0, \forall i \in I, p \in P \cup \{P_0\} \qquad (3.11)$$

$$l_{ip} \geq 0, \forall i \in I, p \in P \qquad (3.12)$$

doi:10.6342/NTU202401914

First of all, we denote P_0 as the first period of the scheduling subproblem, and the parameters with tilde sign (~) show their uncertainties. Constraints (3.2) and (3.3) are related to the state initialization of each subproblem. Constraint (3.2) determined the initial inventory from the given parameter S_i^I, and constraint (3.3) determined the production of each fixed planning period by X_{ij}. Constraint (3.4) guaranteed the mass balance of production, inventory, and demand. Constraint (3.5) and (3.6) cooperate to identify the occurrence of production transition in each period. Constraint (3.7) ensured that for each period, only one of product type could be chosen to produce. Constraint (3.8) determined the sales amount of each period and product. Constraints (3.9) to (3.12) show the domain of each decision variables.

3.3 Robust Optimization

Robust optimization is a relative new branch in mathematical optimization field. In contrast to stochastic optimization, which *starts by assuming the uncertainty has a probabilistic description, RO constructs a solution that is feasible for any realization of the uncertainty in a given set* (Bertsimas et al., 2011). The general formulation of RO is shown in (3.13), where x is a vector of decision variables, u_i is the uncertain parameters, and $\mathcal{U}_i$ is the corresponding uncertainty set for each constraint. After reductions and transformations, the uncertainty set within RO formulation will hopefully being transformed to a new formulation, i.e., *robust counterpart*. An easy

doi:10.6342/NTU202401914

analogy of "robust counterpart" is like a twin of the previous RO formulation. A robust counterpart is less structural (less human readable), but interpretable or solvable for most solver engine, e.g., Gurobi, CPLEX, etc. The transformation process may involve many mathematical theories, we will omit it in this work but can be retrieved from *Lectures on Robust Convex Optimization* (Nemirovski, 2019).

$$\begin{aligned} & min\ f_0(x) \\ & s.t. \quad f_i(x, u_i) \leq 0, \forall u_i \in \mathcal{U}_i, i = 1, \dots, m \end{aligned} \tag{3.13}$$

$\mathcal{U}_i$ can be determined by several approaches (we will only introduce three of them for illustrations):

Ellipsoidal uncertainty

Ben-Tal and Nemirovski (Ben-Tal & Nemirovski, 1999) consider ellipsoidal uncertainty sets, where $\mathcal{U}$ is given as (3.14) and the robust counterpart of this formulation will become (3.15), which is a second-order-cone programming.

$$\mathcal{U} = \{(\boldsymbol{a_1}, \dots, \boldsymbol{a_m}): a_i = a_i^0 + \Delta_i u_i\ , i = 1, \dots, m, \left|\left|u\right|\right|_2 \ \leq\ \rho\} \tag{3.14}$$

$$\begin{aligned} & min\ \boldsymbol{c^T x} \\ & s.t. \quad \boldsymbol{a_i^0 x} \leq b_i - \rho\left|\left|\Delta_i \boldsymbol{x}\right|\right|_2, \forall i \in 1, \dots, m \end{aligned} \tag{3.15}$$

Polyhedral uncertainty

Polyhedral uncertainty can be viewed as a special case of ellipsoidal uncertainty (Ben-Tal & Nemirovski, 1999). Consider a mathematical programming in (3.16), the dual of

doi:10.6342/NTU202401914

the inner maximization problem is (3.17), which can be simplified to a robust counterpart (3.18).

$$\begin{aligned} & min\ \boldsymbol{c^T x} \\ & s.t. \quad max_{\{\boldsymbol{D_i a_i \le d_i}\}} \boldsymbol{a_i^T x} \le b_i, \forall i = 1, \dots, m \end{aligned} \tag{3.16}$$

$$\begin{aligned} & min\ \boldsymbol{p_i^T d_i} \\ & s.t. \quad p_i^T D_i = x \\ & \qquad p_i \le 0 \end{aligned} \tag{3.17}$$

$$\begin{aligned} & min\ \boldsymbol{c^T x} \\ & s.t. \quad p_i^T \le b_i, \quad \forall i = 1, \dots, m \\ & \qquad p_i^T D_i = x, \ \forall i = 1, \dots, m \\ & \qquad p_i \le 0, \quad \forall i = 1, \dots, m \end{aligned} \tag{3.18}$$

Cardinality constrained uncertainty

Bertsimas and Sim (Bertsimas & Sim, 2004) define a family of polyhedral uncertainty sets that encode a budget of uncertainty into cardinality constraints (3.19), and can be transform to a robust counterpart (3.20), which is a tractable linear optimization problem.

$$\begin{aligned} & min\ \boldsymbol{c^T x} \\ & s.t. \quad \sum_j a_{ij} x_j + max_{\{S_i \subseteq J_i\}} \sum_{j \in S_i} \hat{a}_{ij}\, y_j \le b_i, \quad 1 \le i \le m, \\ & \qquad -y_j \le x_j \le y_j, \qquad 1 \le j \le n, \end{aligned} \tag{3.19}$$

doi:10.6342/NTU202401914

$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$$

$$\boldsymbol{y} \geq \boldsymbol{0}$$

$$\begin{aligned}
&min\ \boldsymbol{c^T x} \\
&s.t. \sum_j a_{ij}x_j + z_i\Gamma_i + \sum_j p_{ij} \leq b_i, && \forall i, \\
&\boldsymbol{z_i} + \boldsymbol{p_{ij}} \geq \hat{a}_{ij}\boldsymbol{y_j}, && \forall i, j, \\
&-y_j \leq x_j \leq y_j, && \forall j, \\
&\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \\
&\boldsymbol{p} \geq \boldsymbol{0}, \\
&\boldsymbol{y} \geq \boldsymbol{0}.
\end{aligned} \tag{3.20}$$

In this paper, we will adopt **Cardinality constrained uncertainty set RO** as a guidance for DRL training process.

3.4 Reinforcement learning model

3.4.1 Model design

The agent is modeled using a deep neural network consisting of three hidden layers, each comprising 256 nodes. The rectified linear unit (ReLU) activation function (3.21) is used for these layers:

$$ReLU(x) = max(0, x) \tag{3.21}$$

$$\sigma(x)_m = \frac{e^{x_m}}{\sum_{m=1}^{M} e^{x_m}}, \forall m \in M \tag{3.22}$$

doi:10.6342/NTU202401914

The output of the network employs a softmax function (3.22), resulting in a probability distribution of the discrete actions A. An action, denoted by a, is sampled from A. This action corresponds to x_{ip} in the mathematical model (MM), indicating the product to be scheduled in a particular period. In other words, the product to be produced in each period is determined by a single forward propagation of the state, and a complete schedule is generated through successive propagations.

For instance, as shown in Figure 3 Demonstration of RL episode, illustrates a schedule where the number of products and the planning horizon length are both equal to 4. In each step of the RL agent, a one-hot-encoded output is sampled and assigned to x_{ip}.

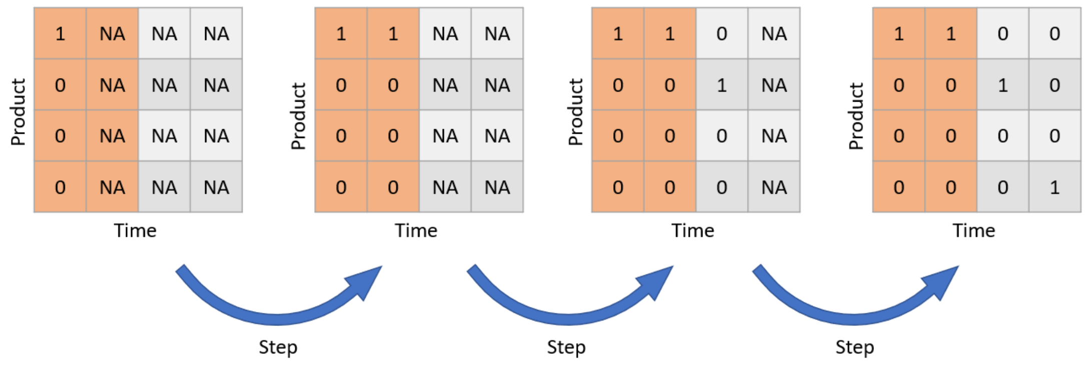
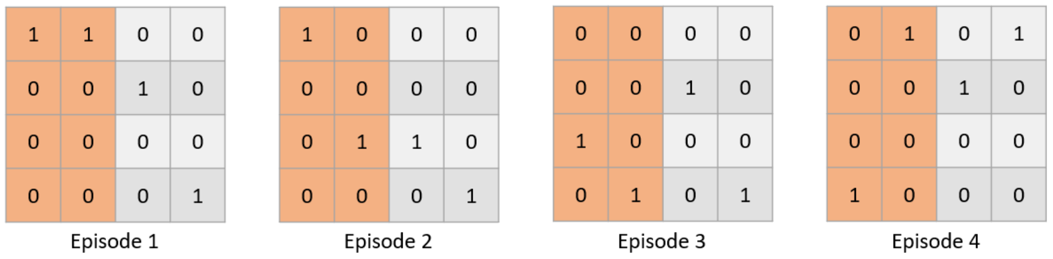


Figure 3 Demonstration of RL episode

An "episode" or a "schedule" is considered complete once all periods have been traversed. According to this schedule, the factory will commence production for one period. After the lapse of a period, the next episode begins, and the process depicted in the last figure is repeated. Notably, in the plan for the next period, there are fixed planning horizons, as exemplified by the first two periods in each episode in Figure 4.

doi:10.6342/NTU202401914



1	1	0	0
0	0	1	0
0	0	0	0
0	0	0	1

Episode 1

1	0	0	0
0	0	0	0
0	1	1	0
0	0	0	1

Episode 2

0	0	0	0
0	0	1	0
1	0	0	0
0	1	0	1

Episode 3

0	1	0	1
0	0	1	0
0	0	0	0
1	0	0	0

Episode 4

Figure 4 Demonstration of an experiment

3.4.2 State definition

The state information, which includes inventory levels, one-hot-encoded current schedules, forecasted demand, estimated stockout, and other parameters of the problem, is provided to the network. This information enables the network to determine the appropriate action to take and assess the current situation.

The inventory level state simply reflects the current inventory of each product, represented as an ordered list of size $|I|$. The current schedules provide the determined schedule information, with unplanned periods filled with zeros. The forecasted demand for each future period is also available. It's worth noting that the demand values used here are assigned with $\widetilde{D}_{ip}$ rather than $\widehat{D}_{ip}$ to prevent data leakage since uncertainties are not revealed during planning. The estimated stockout count for each product type is calculated by subtracting the sum of inventories and production from predicted demands $\tilde{A}_i x_{ip} + s_{ip} - \widetilde{D}_{ip}$. As it aids network convergence, it's also considered part of the state information.

doi:10.6342/NTU202401914

In addition to the state information resulting from the last action, constant parameters C_{ij}^T, C_i^S, C_i^L, V_i and the time counter parameter t are also incorporated. The inclusion of these constant parameters is intended to generalize the agent, enabling it to adapt to different parameter settings in different scenarios.

Lastly, min-max normalization is applied to each type of value, padding zeros are added to the edge periods, and everything is concatenated into a 1D array to serve as the model input, as illustrated in Figure 5.

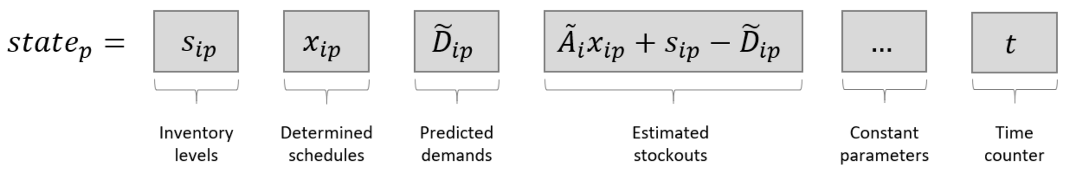


Figure 5 State encoding

3.4.3 Reward definition

The definition of reward is essentially the change in objective value that occurs after a particular action is taken. For instance, if a certain product i is manufactured in a step, the environment will compute the repercussions of this production. If this production will satisfy future demand, it would lead to a decrease in the estimated stockout count, thereby resulting in an increased objective value. In this case, the reward for this step would be the contribution to the improvement in the objective, that is, the original objective value minus the new objective value.

doi:10.6342/NTU202401914

3.5 Proposed OR-Guiding Algorithm

3.5.1 Training Workflow

The workflow of the proposed OR-guiding framework for the described problem is shown as Figure 6:

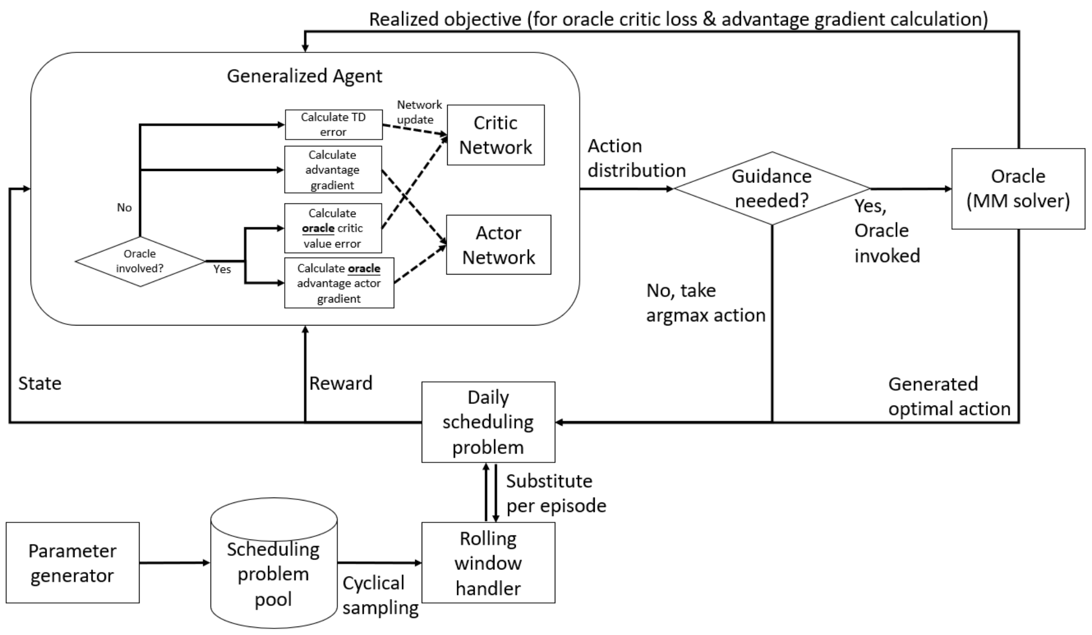


Figure 6 Training workflow

To train an agent that can be applied in variable scenarios, for example, different product selling profit, transition cost, etc., the training set have to include multiple combination of constant parameters instead of single setting. Therefore, we designed a parameter generator that can random sample a parameter set which is reasonable (will be discussed in data simulation section). Given that the training process of RL need to experience the same trajectory for multiple times to actually learn from it (Zhang & Sutton, 2018), we store the generated parameter set as an individual scheduling problem in a space called "scheduling problem pool". In training phase, one of the scheduling

doi:10.6342/NTU202401914

problems in the pool will be cyclically sampled as an experiment (the concept is near to experience replay), and a rolling window handler will transform the experiment into episodes of daily scheduling problems for the agent to experience.

After the reception of daily problem, which is an environment for the agent to interact with, the agent will try to take action on it. Then we will examine the output action distribution, once if we consider guidance is needed, the oracle (MM solver) will be invoked and optimal action substitute the original action.

The agent is composed by A2C method with small modifications: The gradient calculation and update block remains the same while oracle is not involved, else we will calculate “oracle critic value error” and “oracle advantage actor gradient” for critic and actor network, respectively. The detail of those update method will be explained in Section 3.5.2.

3.5.2 Algorithm Implementation

In order to illustrate, we utilize the Reinforcement Learning embedded with Robust Optimization (RLeRO) as an example to demonstrate the use of the OR-guiding framework. The pseudocode of the framework can be found in Table 3.

Before we delve into the algorithm, it's important to understand the update formula of the Advantage-Actor-Critic (A2C) method, which is a prerequisite. We denoted state of step t in episode n as $s_t^{(n)}$ and action as $a_t^{(n)}$,

doi:10.6342/NTU202401914

In the actor-critic method, the actor learns a policy network denoted by θ, which is updated by the policy gradient as shown in Formula (3.15). This gradient is weighted by the Q-value, as determined by the critic network. For instance, if the Q-value is small or even negative, it implies that the given action a is not encouraged in the state s. Consequently, the gradient update corresponding to each state-action pair will influence the probabilities of actions output from the policy network.

The A2C learning algorithm introduces an advantage function $A\left(s_t^{(n)}, a_t^{(n)}\right)$ (3.24) instead of the Q-value function $Q\left(s_t^{(n)}, a_t^{(n)}\right)$ defined in Formula (3.23). In Formula (3.26), which is the approximation of (3.24), we can see that A' is derived by calculating the sum of current reward and discounted state value with a discount rate γ, then minus the previous state value. Ultimately, the critic gradient is derived as shown in Formula (3.27).

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} Q\left(s_t^{(n)}, a_t^{(n)}\right) \cdot \nabla_\theta log p_\theta\left(a_t^{(n)} \middle| s_t^{(n)}\right) \tag{3.23}$$

$$A\left(s_t^{(n)}, a_t^{(n)}\right) = Q\left(s_t^{(n)}, a_t^{(n)}\right) - V\left(s_t^{(n)}\right) \tag{3.24}$$

$$Q\left(s_t^{(n)}, a_t^{(n)}\right) \tag{3.25}$$

$$r^{(n)} + \gamma V\left(s_{t+1}^{(n)}\right) - V\left(s_t^{(n)}\right) = A'\left(s_t^{(n)}, a_t^{(n)}\right). \tag{3.26}$$

$$\nabla_{\theta_V} \sum_{t=1}^{T_n} \frac{1}{T_n} A'\left({s_t}^{(n)}, a_t^{(n)} \middle| \theta_V\right)^2 \tag{3.27}$$

doi:10.6342/NTU202401914

Table 3 Robust optimization embedded A2C learning algorithm framework

```
Requirements:
    • A differentiable policy parameterization $\pi(a|s,\theta_\pi)$.
    • A differentiable state-value parameterization $V(s|\theta_V)$.
    • Select step-size hyper-parameters $0 < \alpha_\pi, \alpha_V \leq 1$.
    • Select guiding-threshold hyper-parameter $\rho > 0$.

1:  **Initialize** the parameters $\theta_V, \theta_\pi$.
2:  **for** $N$ episodes **do**:
3:      Initialize the episode with $s_0$
4:      **for** $T$ steps in episode **do**: $(\forall t \in T)$
5:          Get action probability distribution $p_a$ from current policy $\pi(s_t, \theta_\pi)$
6:          **if** $\text{entropy}(\mathrm{p_a}) > \rho$:
7:              RO invoked and get action $a_t$.
8:              Get realized objective as total return:
                    $f \leftarrow \text{RO\_solver}.realized\_obj()$
9:              Calculate return: $R_t \leftarrow f - f_{<t}$
10:             Take action $a_t$ and observe reward $r_t$ and new state $s_{t+1}$.
11:             Calculate baseline:
                    $b \leftarrow \text{RO\_solver}.fix\_action(\text{argmax}_\mathrm{a}(\mathrm{p_a})).realized\_obj()$
12:             Calculate advantage: $\Delta_t \leftarrow R_t - b$
13:             Calculate partial critic loss: $\kappa_t \leftarrow R_t - V(s_t|\theta_V)$
14:         **else**:
15:             Get action $a_t \leftarrow argmax_a(p_a)$
16:             Take action $a_t$, observe reward $r_t$ and new state $s_{t+1}$
17:             Calculate advantage: $\Delta_t \leftarrow r_t + \gamma V(s_{t+1}|\theta_V) - V(s_t|\theta_V)$
18:             Calculate partial critic loss: $\kappa_t \leftarrow \Delta_t$
19:     Calculate actor loss: $\mathcal{L}(\theta_\pi) \leftarrow -\frac{1}{T}\sum_t^T \Delta_t log\big(\pi(a_t|s_t,\theta_\pi)\big)$
20:     Calculate policy entropy:
            $H\big(\pi(a_t|s_t,\theta_\pi)\big) \leftarrow -\sum_i \pi(a_t|s_t,\theta_\pi)\, log\, \pi(a_t|s_t,\theta_\pi)$
```

doi:10.6342/NTU202401914

```
21:    Update actor: $\theta_\pi \coloneqq \theta_\pi + \alpha_\pi \left(\nabla_{\theta_\pi \mathcal{L}}(\theta_\pi) + \beta \nabla_{\theta_\pi} H\right)$
22:    Calculate critic loss: $\mathcal{L}(\theta_V) = \frac{1}{T}\sum_t^T (\kappa_t)^2$
23:    Update critic: $\theta_V \coloneqq \theta_V + \alpha_V \nabla_{\theta_V} \mathcal{L}(\theta_V)$
Result: trained actor network $\theta_\pi$ and critic network $\theta_V$.
```

The RLeRO training algorithm (Table 3 Robust optimization embedded A2C learning algorithm) inherits the framework from A2C, albeit with some modifications, as underlined in the pseudocode. The differences, marked in bold, will be explained according to their order of execution:

In line 6, we want the RO solver to step in when the agent hesitates among potential actions. This hesitation often stems from two reasons: (1) imminent risks due to unrevealed uncertainty, and (2) the neural network not having learned from similar conditions yet. In both cases, the action distribution outputted by the policy network will have a large entropy. We set up a threshold ρ for this entropy: if the entropy is greater than the threshold, the learning process will enter the "RO guiding block"; otherwise, the original A2C will be employed.

For lines 8 and 9: in the "RO guiding block," the expected return (the sum of remaining rewards) is calculated differently due to the extra information provided by the RO solver. The return R_t is set to the $RO\_realized\_obj_{\geq t}$, which represents the actual return after adopting the RO solution. This value will be used to calculate the *advantage and partial critic loss* in the "RO guiding block."

doi:10.6342/NTU202401914

For lines 11 and 12: the concept of a "Baseline" is central to the A2C algorithm as it aids in the convergence of the actor network. However, in the "RO guiding block," the state-value function is not a suitable baseline because it is unstable and can potentially undermine the information provided by the guiding process. As a substitute, we fix the decision variable for the action in the RO model to $\mathrm{argmax}_{\mathrm{a}}\left(\mathrm{p}_{\mathrm{a}}\right)$ and calculate its realized objective value (denoted as b in the pseudocode). This is a reasonable approach because the two objective values are on the same scale and are thus comparable. The next step involves subtracting b from the return R_t to obtain the advantage value.

The final difference is observed in line 13, where the critic loss compiles partial critic loss from each step (line 22). The partial critic loss κ_t from the "RO guiding" process is defined by $R_t - V(s_t|\theta_V)$, which means the return minus the state-value. The loss function can help the value network θ_V to better approximate the current state-value by learning from the actual (realized) return.

doi:10.6342/NTU202401914

Chapter 4 Numerical Studies

In Chapter 4, we conduct a series of numerical experiments to evaluate the effectiveness of our proposed methods. Initially, we specify the computer and software used in these experiments in Section 4.1. In Section 4.2, we detail the configurations of the parameter generator.

Section 4.3 introduces our models and provides an analysis of their sensitivity against uncertainty parameters. Following this, in Section 4.4, we select suitable scenarios for further exploration. In Section 4.5, we present comparisons of execution time across different models. Lastly, in Section 4.6, we calculate the similarity of Gantt charts to verify the decision-making style of each model.

4.1 Environment Setup

All of the experiments conducted in this study were performed on a computer equipped with an AMD Ryzen 5 5600 6-Core Processor running at 3.50 GHz, supported by 16.0 GB of memory, and utilizing an NVIDIA GeForce RTX 3060 GPU.

On the software side, the experimental codes were implemented in Python, specifically version 3.8.10. Optimization models were solved using Gurobi version 10.0.1 (Gurobi Optimization LLC, n.d.). All neural networks were defined and trained using PyTorch, version 1.13.1.

doi:10.6342/NTU202401914

4.2 Data Simulation

4.2.1 Basic Configurations

The high-level parameters for this study are outlined in Table 4. The length of the experiment horizon and the size of the planning window were selected based on the scale of the problem we needed to investigate further. The number of product types is derived from (Hubbs et al., 2020), which reflects the capabilities of an existing chemical material production machine. The fixed horizon for each window also aligns with the baseline paper, adhering to real-world procedural rules.

Table 4 Experiment parameters

Notation	Description	Setting
$\lvert I \rvert$	Product type count	4
$\lvert \hat{P} \rvert$	Experiment horizon length	90
$\lvert P \rvert$	Planning window size	15
$\lvert F \rvert$	Fixed horizon in each window	7

4.2.2 Parameter Definitions

Instead of defining the precise value for each parameter, we established lower and upper bounds for the parameter generator. The following ranges have been specified (Table 5):

Table 5 Range of parameters

Notation	Description	Setting

doi:10.6342/NTU202401914

$\hat{A}\_lb$	Lower bound of yielding rates	90
$\hat{A}\_ub$	Upper bound of yielding rates	100
$\widehat{D}\_lb$	Lower bound of demands	80
$\widehat{D}\_ub$	Upper bound of demands	120
V_lb	Lower bound of profits	25
V_ub	Upper bound of profits	35

Our parameter generator will draw samples from a uniform distribution to enrich the variety within the scheduling problem pool mentioned in Section 3.4.1 (as listed in the table). Although $\hat{A}_{ip}$ and $\widehat{D}_{ip}$ are generated at the outset and stored in the pool, they remain inaccessible during training and inferencing to prevent data leakage. The only instance these two parameters come into play is during the evaluation phase; agents or optimization models can only access the estimated yielding rates and demands. The notations are listed in Table 6.

Table 6 Sampling methods

Notation	Description	Setting
$\hat{A}_{ip}, \forall i \in I, p \in P$	True yielding rate of each product in each period	$\sim uniform(\hat{A}\_lb, \hat{A}\_ub)$
$\widehat{D}_{ip}, \forall i \in I, p \in P$	True demand of each product in each period	$\sim uniform(\widehat{D}\_lb, \widehat{D}\_ub)$

doi:10.6342/NTU202401914

$V_i, \forall i \in I$	Sales profit of each product	$\sim uniform(V\_lb, V\_ub)$
$S_i^I, \forall i \in I$	Initial inventory of each product	$\sim uniform(\hat{D}\_lb, \hat{D}\_ub)$

The remaining parameters depend on those previously mentioned and are listed in Table 7. C_{ij}^T represents transition costs. The transition process leads to the production of partial unsellable off-grade materials in subsequent periods. Hence, the cost is calculated by multiplying the sales profit of the next product, the yielding rate of the next product, and a constant factor of 0.2 (Hubbs et al., 2020). The definitions of inventory cost C_i^S and stockout cost C_i^L are similar: we assume that the more valuable a product is, the more it costs to store or the higher the cost of stockout.

Uncertain parameters are treated differently in different models. However, all models share the same estimated values for these parameters. In the real-world scheduling problem, while we don't know the actual demand during planning, there exists some estimation of it. For instance, the sum of pre-order tickets may represent future demand. Still, they are subject to change, and we can expect that the further in time the order ticket is, the more likely it is to be modified. Consequently, we designed a demand distortion parameter Δ, which describes the rate of demand value distortion over periods. As illustrated in Figure 7, each $\widetilde{D}_{ip}$ is subject to increased perturbation

doi:10.6342/NTU202401914

as period p increases. This figure shows the relationship between actual and estimated demand, revealing that the estimation error gradually grows due to random noises.

For the yielding rate, which falls between its lower and upper bounds, a naive estimate is its average value.

Table 7 Depending parameters

Notation	Description	Setting
C_{ij}^{T}, $\forall i, j \in I$	Transition costs from product i to product j	$V_j \cdot A_j^* \cdot 0.2$
C_i^S	Inventory costs of products	$V_i \cdot 0.2$
C_i^L	Stockout costs of products	$V_i \cdot 0.3$
$\tilde{A}_i$	Estimated yielding rates	$\frac{\hat{A}\_lb\ +\ \hat{A}\_ub}{2}$
Δ	Demand distortion rate	3
$\widetilde{D}_{ip}$, $\forall i \in I, p \in P$	Estimated demands, fluctuation increased by periods	$\sim uniform\left(\begin{matrix}\left(1-\Delta\cdot\frac{p-P_0}{\lvert P\rvert}\right)\widehat{D}_{ip}, \\ (1+\Delta\cdot\frac{p-P_0}{\lvert P\rvert})\widehat{D}_{ip}\end{matrix}\right)$

doi:10.6342/NTU202401914

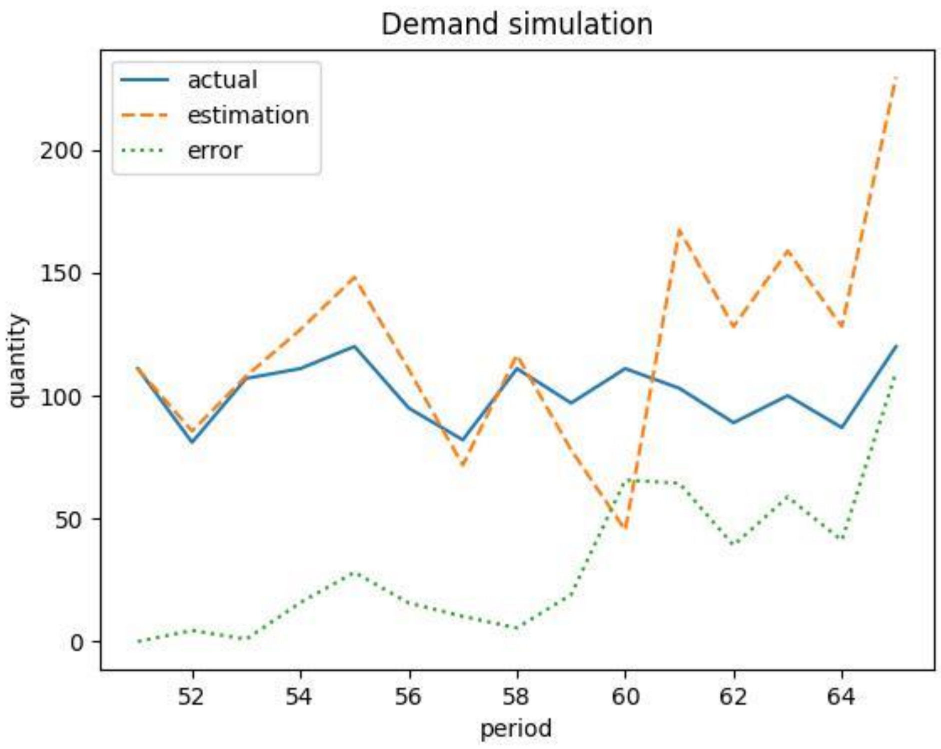


Figure 7 Demand simulation and its errors

4.3 Model Description and Sensitivity Analysis

4.3.1 Model Descriptions

The Perfect Information Deterministic Optimization (PIDO) model, having access to perfect information from the outset, including uncertain parameters, is capable of crafting optimal solutions based on pre-sampled yielding rates and actual demands. This makes its decisions the most effective and efficient. Consequently, it is considered a benchmark against other optimization-based models.

The Expected Value Deterministic Optimization (EVDO) model addresses uncertainty by employing expected values. However, in our use-cases, there are no predefined statistical distributions, so we utilized the mean of the range of uncertain

doi:10.6342/NTU202401914

parameters and distorted demands to estimate the schedules. Since EVDO doesn't account for any other potential data points for each parameter, it is prone to disruptions and this could destabilize its scheduling significantly. This is particularly apparent in scenarios with high distortion rates, where the resulting schedule may fail to meet the actual demand, leading to an increase in stockouts and adversely affecting the stability of the realized objective.

On the other hand, Robust Optimization (RO) is an optimization approach characterized by robust decision-making. It often makes conservative actions to avoid infeasibility in data perturbations. However, such a decision style may lead to a trade-off in some optimality. This trade-off, though, is highly worthwhile in scenarios that necessitate greater planning stability.

4.3.2 Sensitivity Analysis

To select an appropriate fluctuation parameter, we conducted a sensitivity analysis, employing 30 replications (Figure 8 Optimization-based models delta sensitivity analysis). The PIDO model is not influenced by the distortions, which means it can consistently make optimal decisions and achieve the maximum objective value. The EVDO model, only able to observe distorted demands, sees its objective value decrease as delta increases. As for the RO model, which tends to adopt relatively conservative actions, it sacrifices some objective values to achieve more robust results against

doi:10.6342/NTU202401914

demand distortion. The intersection of PIDO and RO occurs at delta = 3.0. This implies that if the demand distortion rate exceeds 3.0, the conservative decisions made by the RO model will ultimately yield better results than the EVDO model.

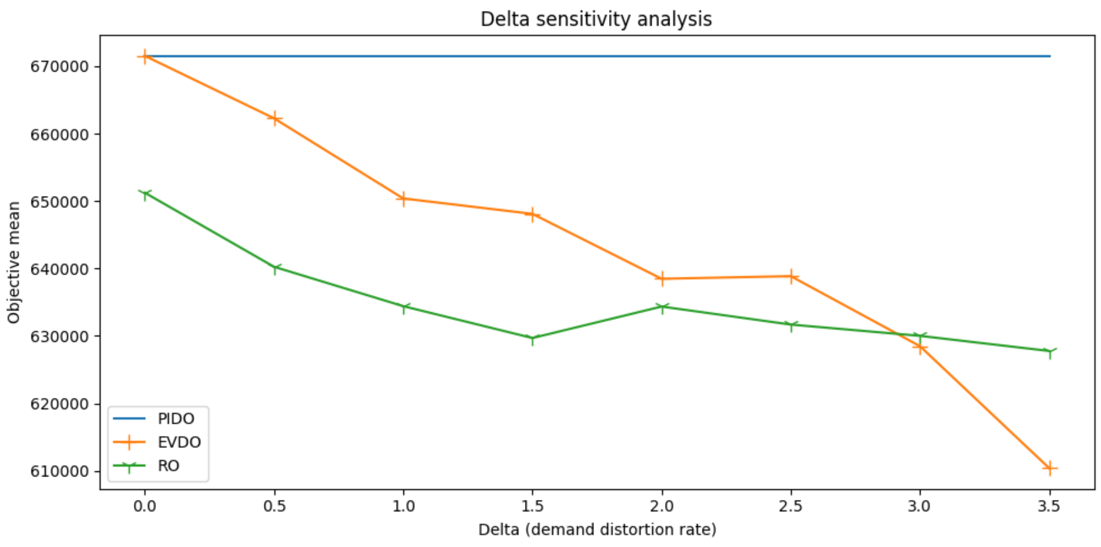


Figure 8 Optimization-based models delta sensitivity analysis

As for the three RL-based models, their performance is depicted in Figure 9. The simple A2C model without any guidance yielded the lowest objective value, regardless of how the demand distortion changed. The two models with guidance, namely A2C+DO and RLeRO, didn't show big differences in objective value when the distortions were small. However, when the distortion rate increased to 2.0, the A2C

doi:10.6342/NTU202401914

model guided by RO (named RLeRO) demonstrated stability in objective values.

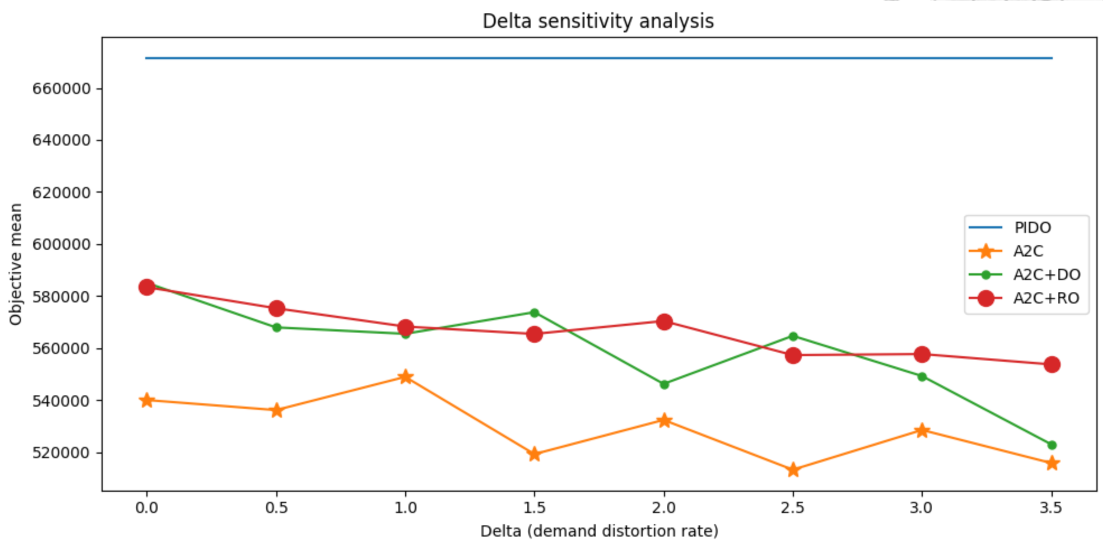


Figure 9 RL-based models delta sensitivity analysis

Both the optimization and RL approaches showed intersections at the value of delta = 3.0, so we will conduct further investigations in these scenarios in section 4.4. To explore the influence of all uncertain variables in the RLeRO model, we also conducted a two-dimensional sensitivity analysis with 5 replications, as shown in Table 8. The results reveal a declining trend in objective values as both Δ and the yielding rate sample range (YRSR) increase.

Table 8 Two-dimensional (YRSR & Δ) sensitivity analysis

YRSR\Δ	0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
0	583,413 (3,141)	575,313 (3,460)	568,285 (4,679)	565,447 (5,483)	570,411 (6,578)	557,310 (7,318)	557,715 (7,920)	553,939 (9,446)
0.1	580,249 (3,222)	575,223 (3,660)	567,821 (4,933)	565,498 (5,567)	556,501 (6,829)	556947 (7,356)	557,601 (8,913)	554,173 (9,862)
0.2	578,952 (3,349)	575,106 (3,725)	566,366 (4,968)	564,997 (5,633)	557,063 (7,217)	557,211 (7,542)	555,408 (9,298)	552,382 (9,863)
0.3	579,431 (3,442)	575,006 (3,862)	565,264 (5,098)	565,510 (5,919)	556,880 (7,250)	556,885 (7,685)	556,019 (9,348)	551,766 (9,930)
0.4	578,407	576,757	570,845	565,592	556,751	556,631	555,807	550,708

doi:10.6342/NTU202401914

	(3,449)	(4,264)	(5,194)	(5,998)	(7,292)	(7,881)	(9,394)	(9,985)

4.4 Solution Value Analysis

4.4.1 Descriptive Statistics

As mentioned in Section 4.3, we selected high demand distortion cases where RO outperformed EVDO and RLeRO surpassed A2C + DO, demonstrating their robustness.

The results for the three optimization-based models are shown in Table 9 Optimization-based models' objective statistics. The RO model had better objective and standard deviation values than EVDO. As expected, the improvement in both mean and standard deviation becomes more pronounced as the distortion parameter increases. We chose the edge case primarily for easier observation.

Table 9 Optimization-based models' objective statistics

Model	Realized obj. Mean	Realized obj. Std.	Gap on obj. mean	Note
PIDO	671,540	0	-	Benchmark
EVDO	628,473	29,562	6.41%	-
RO	630,003	23,875	6.18%	Improve 19.23% in std. versus EVDO

The comparison of the three RL-based models is shown in Table 10 RL-based models' objective statistics. Without specific design of network structure and hyper-parameters, the A2C method achieved about 81% of the benchmark's performance.

doi:10.6342/NTU202401914

However, its performance in terms of standard deviation is not ideal, with an increase of nearly 30% compared to the EVDO.

The proposed guiding framework in this paper aims to improve learning effectiveness in the early stages of model training and in situations with dilemmas, by using mathematical optimization to guide subproblems. In the first experiment, DO was used as the guiding engine, resulting in a 4% increase in the realized objective value compared to A2C, reaching 82% of the benchmark, and a 11% reduction in standard deviation.

For application scenarios that require higher stability, both A2C and A2C+DO guiding still need further improvement in terms of standard deviation. Therefore, the second experiment aimed to bring about a change in decision-making style through the guiding process during learning. We used RO as the guiding engine and observed that the model indeed learned to adopt a more conservative strategy during inference, which achieved a 9% reduction in standard deviation. This final model is named as Reinforcement Learning embedded with Robust Optimization, abbreviated as RLeRO.

Table 10 RL-based models' objective statistics

Model	Realized Obj. Mean	Realized Obj. Std.	Improvement on obj. mean	Improvement on obj. std	Note

doi:10.6342/NTU202401914

A2C	528,541	38,426	-	-	Baseline
<u>A2C + DO</u>	549,265	34,013	3.92%	11.48%	Better than baseline
<u>RLeRO</u> <u>(A2C + RO)</u>	557,715	30,650	5.52%	20.23%	Better than A2C + DO

4.4.2 Evaluation Metrics

For research with probability distribution assumptions, the "expected value of perfect information (EVPI)" and the "value of stochastic solution (VSS)" can be used to identify the value of stochastic programming. However, all of the experiments mentioned above were conducted without assuming statistical distributions, meaning that VSS is not applicable in our cases. Since our major discussion revolves around robustness against parameter fluctuation, we borrow the idea of VSS and substitute it with a metric known as "price of robustness (PR)" (Chassein & Goerigk, 2016).

The main concept of PR is to compute the distance between the EVDO and robust solution (RS) in terms of objective value (4.1). In our research case, we calculate the mean PR of 30 replications at different distortion rates.

In real-world situations, we cannot know the demand fluctuation, i.e., the distortion coefficient, in advance. Thus, we analyze the price of robustness in different scenarios to determine the best application of robust models. The results are shown in

doi:10.6342/NTU202401914

Figure 10: We observe a negative trend in the price of robustness for both RLeRO and RO models, which indicates that the sacrifice of applying a robust solution diminishes as the fluctuation rate increases. In cases with high demand fluctuation, it is even more beneficial to adopt conservative policies.

$$PR = EVDO - \mathrm{RS} \tag{4.1}$$

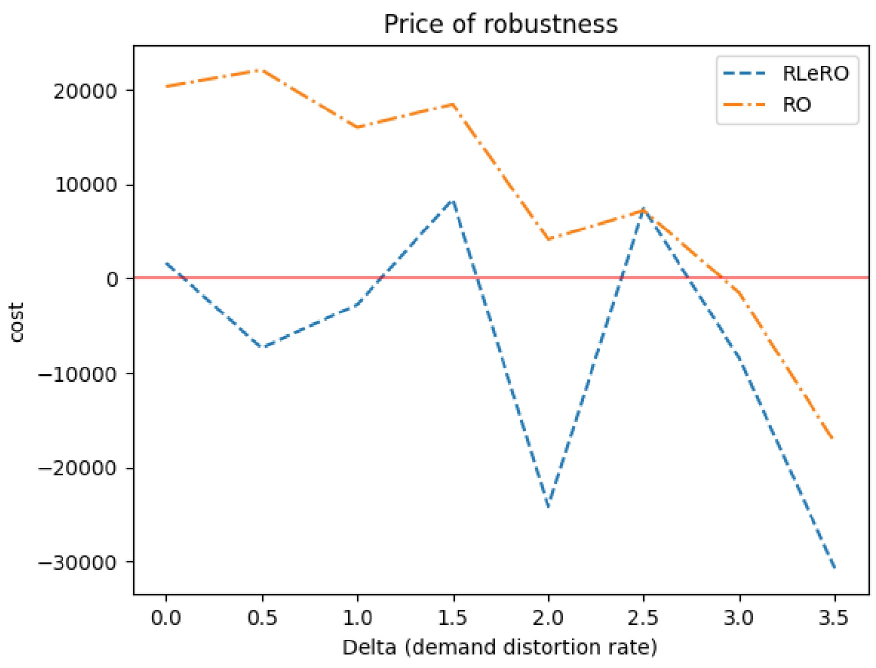


Figure 10 Price of robustness for Robust models

The EVPI is calculated by subtracting the PIDO from the wait-and-see (WS) objective value (2), which represents how much a decision-maker would be willing to pay for perfect information. The WS models are those we want to evaluate. In our case, we want to show that the robust models are less needed of accurate (perfect) information, i.e., low EVPI values.

In our case, we separate the calculation result into two figures - optimization-based (Figure 11) and RL-based (Figure 12) - to compare them to their respective baseline

doi:10.6342/NTU202401914

models. We can observe that in the optimization-based group, the EVPI value of DO surpasses that of RO at high demand distortion rates. For the RL-based group, RLeRO has the lowest value compared to the others. These results demonstrate that the robust models are more resilient to imperfect information.

$$EVPI = PIDO - \text{WS} \tag{4.2}$$

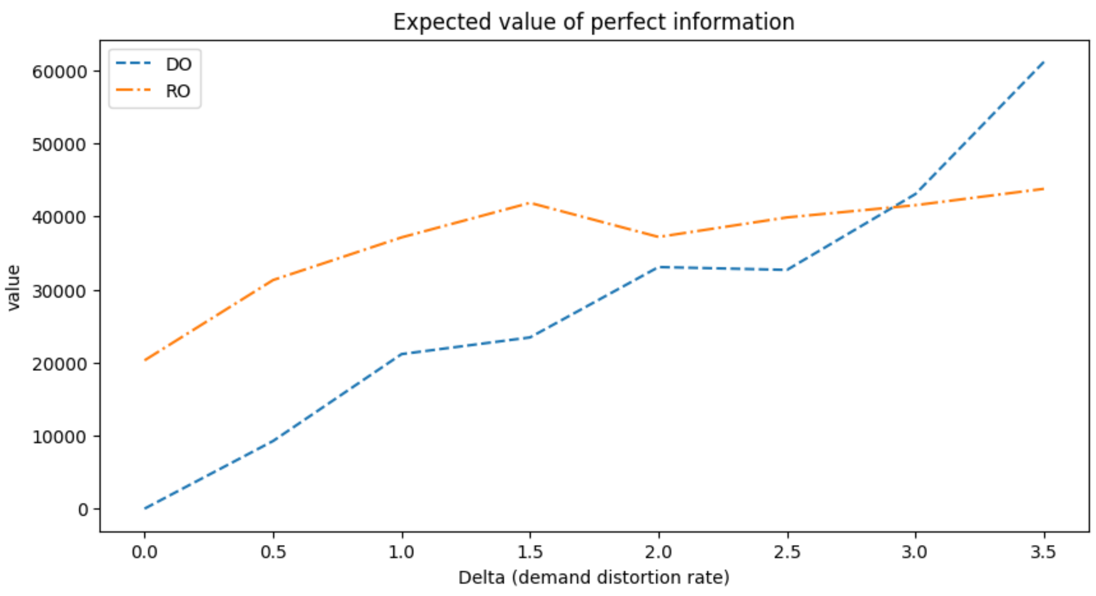


Figure 11 EVPI of Optimization-based models

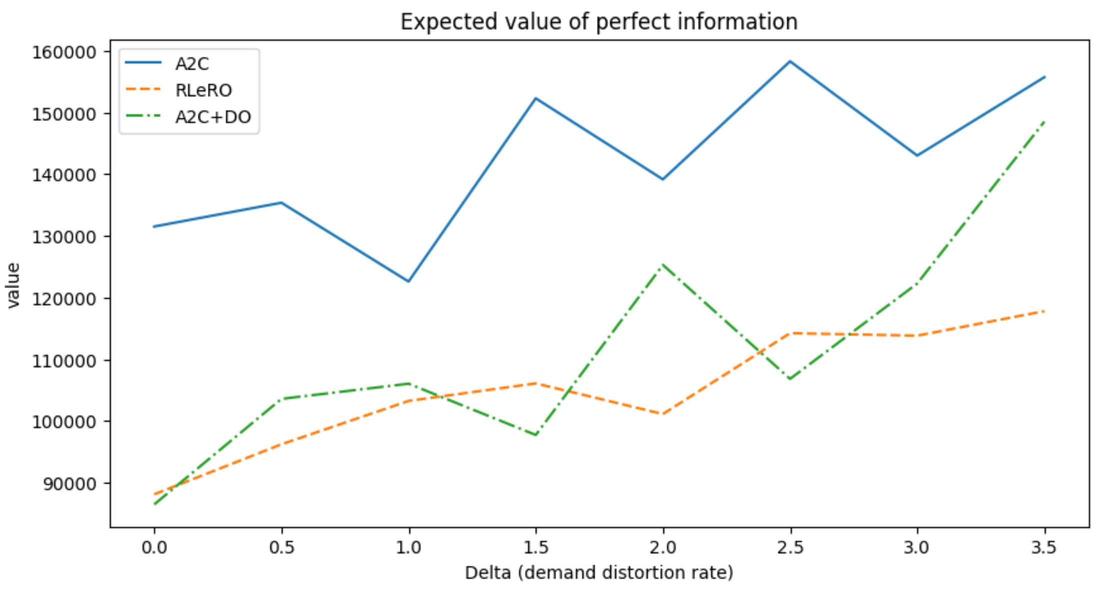


Figure 12 EVPI of RL-based models

doi:10.6342/NTU202401914

4.5 Execution Time Analysis

In traditional optimization methods, there is no distinction between training and inference. The process begins with constructing a mathematical model for the scheduling problem, which is then solved using an optimizer to obtain the optimal solution. Each rescheduling event requires solving the problem with different inputs, and the required time grows quickly with the size of the problem. Conversely, in RL applications, the initial step involves model training, which is the most computationally intensive part of the entire process. Solving a scheduling problem simply necessitates running inference once, a process that takes relatively less time compared to training. This gives RL an advantage in scenarios requiring frequent rescheduling. Moreover, as long as state information is effectively designed for input parameters, the same model can be used for inference and scheduling across different parameter settings.

Figure 13 presents a comparison of the time consumption among various models for different problem sizes (planning horizon lengths). As the bottleneck for RL lies in the training phase, the training time serves as the comparison metric for the solver. Each data point denotes the average result from 20 experiments. It can be observed that both the deterministic model and the robust optimization model display an exponential-like growth in solving time, failing to converge within the set time limit of 3000 seconds when the planning horizon exceeds 60. The general A2C method, however, has a

doi:10.6342/NTU202401914

significantly slower growth curve and surpasses traditional optimization methods when the problem size approaches 50. Our proposed guiding framework, which solves and guides the model's uncertain subproblems during the training process, necessitates more time in the training phase compared to a simple A2C model. However, compared to models with exponential growth, this computational burden is still within an acceptable range.

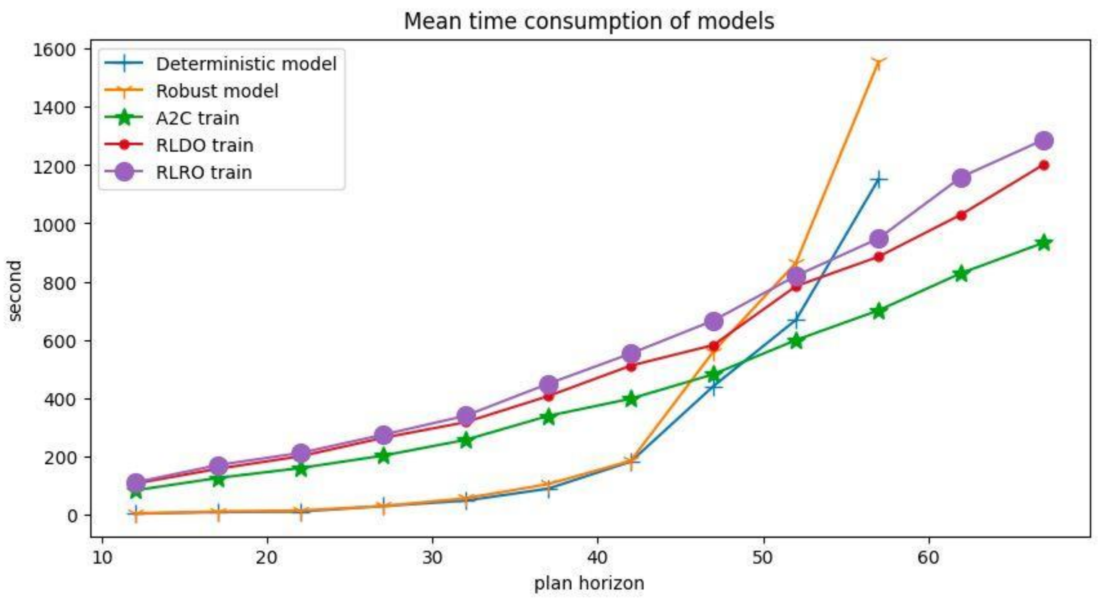


Figure 13 Bottleneck time consumption of each model

Regarding the inference time for the RL model, which refers to the time consumed during the forward propagation of state information through the network, it is extremely minimal when compared to RL training and MM solving. The mean inference time consumption across different planning horizons, calculated from 30 replications, is shown in Table 11. From these results, we can observe that the differences in inference time across different planning horizons are small enough to be negligible.

doi:10.6342/NTU202401914

Table 11 Inference time of RL-based models

Model	|P| = 10	|P| = 20	|P| = 40	|P| = 60
A2C	0.09s (0.02)	0.09s (0.03)	0.09s (0.02)	0.11s (0.02)
A2C+DO	0.09s (0.02)	0.09s (0.03)	0.09s (0.01)	0.10s (0.02)
RLeRO (A2C + RO)	0.09s (0.02)	0.09s (0.02)	0.08s (0.02)	0.09s (0.02)

4.6 Gantt Similarity Analysis

To examine the differences between models, we calculate a "Gantt similarity" score for each combination of models based on 30 replications (Table 12). The similarity score is determined by comparing the production decisions in segments of the schedule, where the proportion of matching segments represents the similarity.

We observe a significant similarity gap between RL-based and optimization-based methods. However, when guiding methods are introduced, this gap narrows considerably.

Table 12 Gantt similarities between models

	EVDO	RO	A2C	A2C + DO	RLeRO (A2C + RO)
EVDO	1	93.55% (8.24)	38.71% (15.32)	74.19% (10.56)	70.96% (8.63)
RO	-	1	45.05%	64.51%	80.64%

doi:10.6342/NTU202401914

			(5.70)	(14.37)	(7.01)
A2C	-	-	1	45.16% (7.55)	32.25% (4.98)
A2C + DO	-	-	-	1	73.52% (8.17)
RLeRO (A2C + RO)	-	-	-	-	1

Figure 14 and Figure 15 each represent a Gantt chart from the RO and RLeRO models, respectively, within the same experiment simulation. It is evident that the schedules bear similarity, indicating that the latter model has indeed learnt from the RO guiding phase and is inclined towards conservative actions. By contrast, the simple A2C model (Figure 16) has lesser commonality with the RO Gantt chart. This comparison also corroborates that the RLeRO model effectively mimics the decision-making style of the RO model.

doi:10.6342/NTU202401914

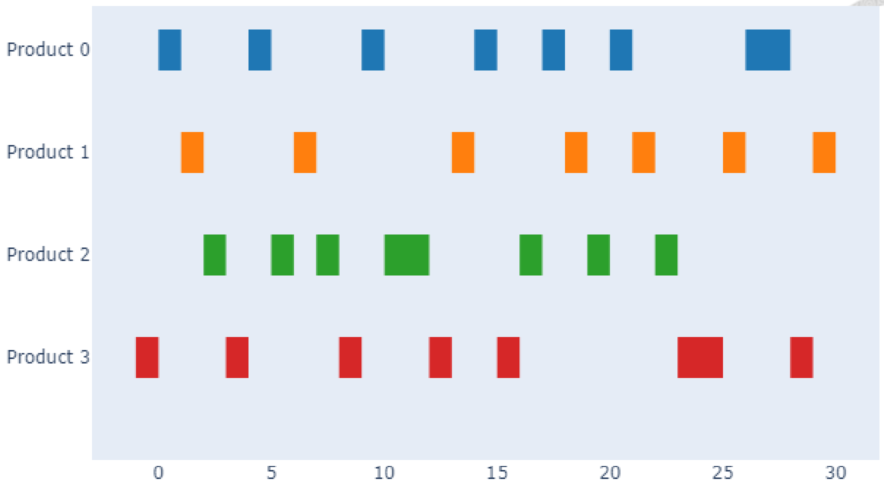


Figure 14 Gantt of RO solution in scenario 1

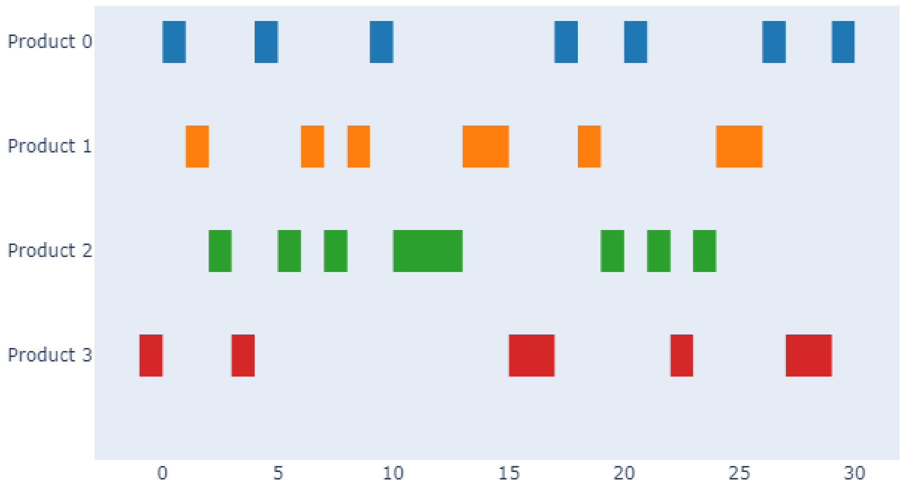


Figure 15 Gantt of RLeRO solution in scenario 1

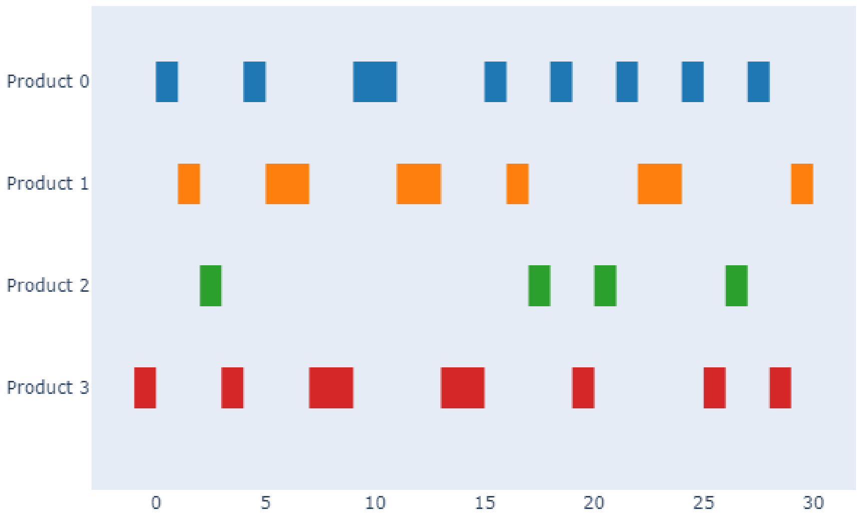


Figure 16 Gantt of A2C solution in scenario 1

doi:10.6342/NTU202401914

Chapter 5 Conclusion and Future Works

5.1 Conclusion

This paper presents a guiding framework for training reinforcement learning models, specifically for the dynamic scheduling of a single-stage multi-product reactor. The framework was motivated by the limitations observed in previous research: *"There is no guarantee of optimality with policy gradient methods, as the reinforce algorithm can converge to local optimality.* (Sutton et al., 1999)" As such, we initially attempted to guide the training phase, and as expected, our proposed approach improved decision quality in terms of objective value and stability, compared to simple A2C methods. To further enhance robustness against parameter distortions, we adopted robust optimization as a guiding engine, which enabled a more conservative decision-making style. The effectiveness of these improvements was validated through experimental results, where we compared different models using the "Price of Robustness" metric, confirming the enhanced robustness of the RO and RLeRO methods.

5.2 Future Works

We have made strides in integrating DRL and optimization methods, addressing the research gaps highlighted in baseline paper. Nonetheless, there remains ample room for further exploration in this field. For instance, future research could attempt to

doi:10.6342/NTU202401914

incorporate other guiding methods during the training phase, or modify the gradient and loss functions for a better synergy between RL and optimization methods.

We also aim to broaden the scope of our training framework beyond scheduling problems. One potential direction is the development of a converter that can translate a mathematical model into an RL environment. This could facilitate the application of our proposed framework in a wider range of problem contexts.

The lack of comparative studies on different RL guiding methodologies is noteworthy. Future research could utilize approaches like "decaying perfect information" as a baseline for more in-depth analysis and evaluation of these methods.

doi:10.6342/NTU202401914

References

Andriushchenko, M., & Flammarion, N. (2020). Understanding and Improving Fast Adversarial Training. *Advances in Neural Information Processing Systems*, *33*, 16048–16059. https://proceedings.neurips.cc/paper/2020/hash/b8ce47761ed7b3b6f48b583350b7f9e4-Abstract.html

Ben-Tal, A., & Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations Research Letters*, *25*(1), 1–13. https://doi.org/10.1016/S0167-6377(99)00016-4

Bertsimas, D., Brown, D. B., & Caramanis, C. (2011). Theory and Applications of Robust Optimization. *SIAM Review*, *53*(3), 464–501. https://doi.org/10.1137/080734510

Bertsimas, D., & Sim, M. (2004). The Price of Robustness. *Operations Research*, *52*(1), 35–53. https://doi.org/10.1287/opre.1030.0065

Bonfill, A., Bagajewicz, M., Espuña, A., & Puigjaner, L. (2004). Risk Management in the Scheduling of Batch Plants under Uncertain Market Demand. *Industrial &amp; Engineering Chemistry Research*, *43*(3), 741.

doi:10.6342/NTU202401914

Bonfill, A., Espuña, A., & Puigjaner, L. (2005). Addressing Robustness in Scheduling Batch Processes with Uncertain Operation Times. *Industrial & Engineering Chemistry Research*, *44*(5), 1524–1534. https://doi.org/10.1021/ie049732g

Chang, J., Yu, D., Hu, Y., He, W., & Yu, H. (2022). Deep Reinforcement Learning for Dynamic Flexible Job Shop Scheduling with Random Job Arrival. *Processes*, *10*, 760. https://doi.org/10.3390/pr10040760

Chassein, A., & Goerigk, M. (2016). Performance Analysis in Robust Optimization. In M. Doumpos, C. Zopounidis, & E. Grigoroudis (Eds.), *Robustness Analysis in Decision Aiding, Optimization, and Analytics* (Vol. 241, pp. 145–170). Springer International Publishing. https://doi.org/10.1007/978-3-319-33121-8_7

Grossmann, I. E., Apap, R. M., Calfa, B. A., García-Herreros, P., & Zhang, Q. (2016). Recent advances in mathematical programming techniques for the optimization of process systems under uncertainty. *Computers & Chemical Engineering*, *91*, 3–14. https://doi.org/10.1016/j.compchemeng.2016.03.002

Gupta, D., & Maravelias, C. T. (2016). On deterministic online scheduling: Major considerations, paradoxes and remedies. *Computers & Chemical Engineering*, *94*, 312–330. https://doi.org/10.1016/j.compchemeng.2016.08.006

Gurobi Optimization LLC. (n.d.). *Gurobi Optimizer Reference Manual*.

doi:10.6342/NTU202401914

Han, D., Kozuno, T., Luo, X., Chen, Z., Doya, K., Yang, Y., & Li, D. (2022, March 16). *Variational oracle guiding for reinforcement learning*.

Hoek, R. (2020). Research opportunities for a more resilient post-COVID-19 supply chain – closing the gap between research findings and industry practice. *International Journal of Operations & Production Management*, *ahead-of-print*. https://doi.org/10.1108/IJOPM-03-2020-0165

Hu, Z., Ramaraj, G., & Hu, G. (2020). Production planning with a two-stage stochastic programming model in a kitting facility under demand and yield uncertainties. *International Journal of Management Science and Engineering Management*, *15*(3), 237–246. https://doi.org/10.1080/17509653.2019.1710301

Hubbs, C. D., Li, C., Sahinidis, N. V., Grossmann, I. E., & Wassick, J. M. (2020). A deep reinforcement learning approach for chemical production scheduling. *Computers & Chemical Engineering*, *141*, 106982. https://doi.org/10.1016/j.compchemeng.2020.106982

Janak, S. L., Floudas, C. A., Kallrath, J., & Vormbrock, N. (2006). Production Scheduling of a Large-Scale Industrial Batch Plant. II. Reactive Scheduling. *Industrial & Engineering Chemistry Research*, *45*(25), 8253–8269. https://doi.org/10.1021/ie0600590

doi:10.6342/NTU202401914

Janak, S. L., Lin, X., & Floudas, C. A. (2007). A new robust optimization approach for scheduling under uncertainty: II. Uncertainty with known probability distribution. *Computers & Chemical Engineering*, *31*(3), 171–195. https://doi.org/10.1016/j.compchemeng.2006.05.035

Jia, Z., & Ierapetritou, M. (2006a). Generate Pareto Optimal Solutions of Scheduling Problems Using Normal Boundary Intersection Technique. *Computers & Chemical Engineering*, *1*, 268–280.

Jia, Z., & Ierapetritou, M. (2006b). Uncertainty analysis on the righthand side for MILP problems. *AIChE Journal*, *52*, 2486–2495. https://doi.org/10.1002/aic.10842

Jung, J. Y., Blau, G., Pekny, J. F., Reklaitis, G. V., & Eversdyk, D. (2004). A simulation based optimization approach to supply chain management under demand uncertainty. *Computers & Chemical Engineering*, *28*(10), 2087–2106. https://doi.org/10.1016/j.compchemeng.2004.06.006

Lee, C.-Y., Huang, Y.-T., & Chen, P.-J. (2024). Robust-optimization-guiding deep reinforcement learning for chemical material production scheduling. *Computers & Chemical Engineering*, *187*, 108745. https://doi.org/10.1016/j.compchemeng.2024.108745

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of Machine Scheduling Problems. In P. L. Hammer, E. L. Johnson, B. H. Korte, & G. L. Nemhauser (Eds.), *Annals of Discrete Mathematics* (Vol. 1, pp. 343–362). Elsevier. https://doi.org/10.1016/S0167-5060(08)70743-X

Li, J., Koyamada, S., Ye, Q., Liu, G., Wang, C., Yang, R., Zhao, L., Qin, T., Liu, T.-Y., & Hon, H.-W. (2020). *Suphx: Mastering Mahjong with Deep Reinforcement Learning* (arXiv:2003.13590). arXiv. http://arxiv.org/abs/2003.13590

Li, Z., & Ierapetritou, M. (2008). Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, *32*, 715–727. https://doi.org/10.1016/j.compchemeng.2007.03.001

Lin, X., Janak, S. L., & Floudas, C. A. (2004). A new robust optimization approach for scheduling under uncertainty: I. Bounded uncertainty. *Computers & Chemical Engineering*, *28*(6), 1069–1085. https://doi.org/10.1016/j.compchemeng.2003.09.020

Mihoubi, B., Bouzouia, B., & Gaham, M. (2021). Reactive scheduling approach for solving a realistic flexible job shop scheduling problem. *International Journal of Production Research*, *59*(19), 5790–5808. https://doi.org/10.1080/00207543.2020.1790686

doi:10.6342/NTU202401914

Nemirovski, A. (2019). *Lectures on Robust Convex Optimization* [Dataset]. https://doi.org/10.1287/e356790b-ddcc-4920-a645-a2d08c6334bb

Petrovic, D., & Duenas, A. (2006). A fuzzy logic based production scheduling/rescheduling in the presence of uncertain disruptions. *Fuzzy Sets and Systems*, *157*(16), 2273–2285. https://doi.org/10.1016/j.fss.2006.04.009

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., & Liang, P. (2019). *Adversarial Training Can Hurt Generalization* (arXiv:1906.06032). arXiv. http://arxiv.org/abs/1906.06032

Riedmiller, S., & Riedmiller, M. (n.d.). *A neural reinforcement learning approach to learn local dispatching policies in production scheduling*.

Sahinidis, N. V. (2004). Optimization under uncertainty: State-of-the-art and opportunities. *Computers & Chemical Engineering*, *28*(6–7), 971–983. https://doi.org/10.1016/j.compchemeng.2003.09.017

Sand, G., & Engell, S. (2004). Modeling and solving real-time scheduling problems by stochastic integer programming. *Computers & Chemical Engineering*, *28*(6), 1087–1103. https://doi.org/10.1016/j.compchemeng.2003.09.009

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach,

doi:10.6342/NTU202401914

M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems, 12*. https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html

Zhang, S., & Sutton, R. S. (2018). *A Deeper Look at Experience Replay* (arXiv:1712.01275). arXiv. https://doi.org/10.48550/arXiv.1712.01275

doi:10.6342/NTU202401914