World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:17, No:8, 2023

Deep Reinforcement Learning for Optimal Decision-making in Supply Chains

Nitin Singh, Meng Ling, Talha Ahmed, Tianxia Zhao, Reinier van de Pol

Abstract—We propose the use of Reinforcement Learning (RL) as a viable alternative for optimizing supply chain management, particularly in scenarios with stochasticity in product demands. RL's adaptability to changing conditions and its demonstrated success in diverse fields of sequential decision-making make it a promising candidate for addressing supply chain problems. We investigate the impact of demand fluctuations in a multi-product supply chain system and develop RL agents with learned generalizable policies. We provide experimentation details for training RL agents and a statistical analysis of the results. We study generalization ability of RL agents for different demand uncertainty scenarios and observe superior performance compared to the agents trained with fixed demand curves. The proposed methodology has the potential to lead to cost reduction and increased profit for companies dealing with frequent inventory movement between supply and demand nodes.

Keywords—Inventory Management, Reinforcement Learning, Supply Chain Optimization, Uncertainty.

I. Introduction

SUPPLY chain optimization is critical for almost every company that deals with frequent inventory movement between supply and demand nodes. Optimized movement of inventory from one node to another in response to the product demand can potentially lead to a reduction in the overall costs and an increased profit. Typically, companies employ standard procedures such as fixed-reorder policies for supply chain management [1]. Even though such heuristic policies are simplistic, they are commonly used in industrial applications: fixed policies offer ease of understanding, tractability and can be easily utilized by supply chain professionals. However, fixed policies have limitations in applicability such as: *(i)* the frequent recommendation of suboptimal solutions, and *(ii)* the inability to perform well in large networks and complex stochastic environments, such as those influenced by demand fluctuations, cost variations, etc. [2].

Supply chain optimization is, in essence, a complex sequential decision-making problem with the objective of maximizing profit and minimizing monetary or product loss within a given time horizon. Overall supply chain optimization requires the optimization of each sequential step, e.g., the transfer of product from supply to distribution warehouses. Advancements in Deep Reinforcement Learning (DRL), a research area in Reinforcement Learning (RL), have shown great promise in recent years across different fields of sequential decision-making, e.g., robotics, AI game playing agents, automated stock trading, process control,and self-driving cars etc. [3]-[5]. The success of RL in these fields suggests the potential for the future application of RL-based optimal decision-making models to other dynamic systems where an adaptive control is required [6]-[8]. We hypothesize that RL can be a viable alternative for supply chain optimization problems, particularly in scenarios with stochasticity in product demands. The main benefit of RL over most classically-used methods is its adaptability; RL is successful at accommodating the varying conditions often present in supply chains [9]. A generic policy can be learnt that adapts well to variability in supply chain environment such as demand fluctuations.

The primary objective of this research is to develop RL agents with generalizable and robust policies. We investigate the impact of demand fluctuations in a multi-product supply chain environment that comprises of a single supply node delivering inventory to two customer nodes (Fig. 1, Section III). Products are shipped on a daily basis, and there is an uncertainty associated with the product demands. We study sharp (single day) fluctuations in product demand and develop RL agents for three analogous sub-scenarios. In the experimental scenarios examined, a specific training protocol is implemented for the agents. During the training process, the agents are exposed to a variety of demand curves, selected from a pre-prepared set. The set of demand curves is carefully crafted to cover the entire spectrum of demand disruptions that the agents are expected to encounter during subsequent testing. Results indicate that agents trained with this protocol are generalizable to test scenarios and surpass the performance of baseline agents, i.e., RL agents trained with fixed demand curves.

This paper is organized as follows. Section II presents related works in the field of supply chain optimization. In Section III, we describe the supply chain network and present how the supply chain is modeled as a Markov Decision Process. We discuss basics of RL and our training methodology to design adaptive agents in Section IV. Results are presented in Section V, followed by conclusions and suggestions for future research in Section VI.

II. Literature Survey

In this section, we briefly discuss some recently published available literature relevant to supply chain optimization

Nitin Singh is with Shell (India), Operations Research team, Bangalore, India (phone: +916239526184; e-mail: nitin.n.singh@shell.com).

Meng Ling and Tianxia Zhao are with the Shell (USA), Deep Learning and AI team, Houston, TX, USA (phone: 832-248-3617; e-mail: meng.ling@shell.com, tina.t.zhao@shell.com).

Talha Ahmed is with the University of Tennessee, Knoxville, TN 37916 USA (phone: 865-978-2038; e-mail: tahmed4@vols.utk.edu).

Reinier van de Pol is with Shell (Netherlands), Operations Research team, Amsterdam, Netherlands (e-mail: reinier.vandepol@shell.com).

through RL. The work presented in [6] utilizes DRL algorithms such as PPO to solve the supply chain inventory management problem under three distinct scenarios representative of different supply chain networks, comparing the performance of each scenario to standard inventory management policies. The authors in [11] investigate the impact of three different RL methods on the optimization of safety stock in a linear chain of independent agents. A key feature of presented RL based approach is that it can simultaneously optimize both safety stock level and order quantity parameters of an inventory policy. Reference [10] models the supply chain optimization problem as a Markov Decision Process to design the environment and utilizes two different RL algorithms namely SARSA and REINFORCE alongside the conventional static policy to obtain an optimal set of inventory management actions.

Reinforcement learning based optimization approaches have also been employed to deal with complex supply chain scenarios involving uncertain events. For multi-product inventory problems, [12] uses the advantage actor critic (A2C) and deep Q-network (DQN) algorithms with quantized action spaces for optimization. In this work, authors consider realistic business goals such as minimization of wastage as the optimization objective and also show that the learned policy can be transferred with minimal training to a supply chain with different number of products. Furthermore, a more complicated supply chain is considered in [9], with optimization depending on the additional factors of demand uncertainty and multi-level supply chains. For optimization, a deep RL algorithm named PPO [15] is deployed to solve the problem for a wide array of uncertainities i.e. uncertain, regular and seasonal demands and constant or stochastic lead times. A similar approach in [13] presents two DRL based methods to solve multi-period capacitated supply chain optimization problem under demand uncertainty while considering both continuous and discrete action space. Based on the literature reviewed, it appears that deep RL algorithms may offer certain advantages over traditional inventory management algorithms in dealing with supply chain optimization problems.

III. Problem Description

In this section we present the supply chain system and discuss modelling details of optimal sequential decision-making through Markov Decision Process framework.

A. Supply Chain Network

Due to confidentiality considerations, it is not possible to provide precise numerical data for the supply chain. Instead, we present normalized or scaled data where necessary.

Fig. 1 shows a schematic of the supply chain. The network consists of a single factory (F) serving two customers: C_1 and C_2. The factory produces 3 products each of two categories: P_1 and P_2 and ships them to C_1 and C_2 respectively. There are six products in total: P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, P_{23}, with the first three delivered to C_1 and the later three delivered to C_2. A separate storage warehouse and a separate transport is allocated for moving each product between factory and customers. The warehouses start with certain initial stock levels and have associated minimum and maximum storage capacities (in kg). Carrier transports for each product also have a given maximum load capacity. For simplicity, we assume fixed daily production (kg/day) of each product and no transit times associated with the product movement in the network. We have associated costs including production costs, storage costs and transportation costs measured in USD/kg. All costs are assumed to be constant. Initially, we consider a fixed daily demand for the products (kg/day), but we later investigate demand fluctuations as well. The shipping decisions from factory to customers are made on a daily basis. Each period requires determining the amount, in kilograms, of each product to be stored at warehouses and the amount shipped to customers. The objective is to maximize the annual cumulative profit.

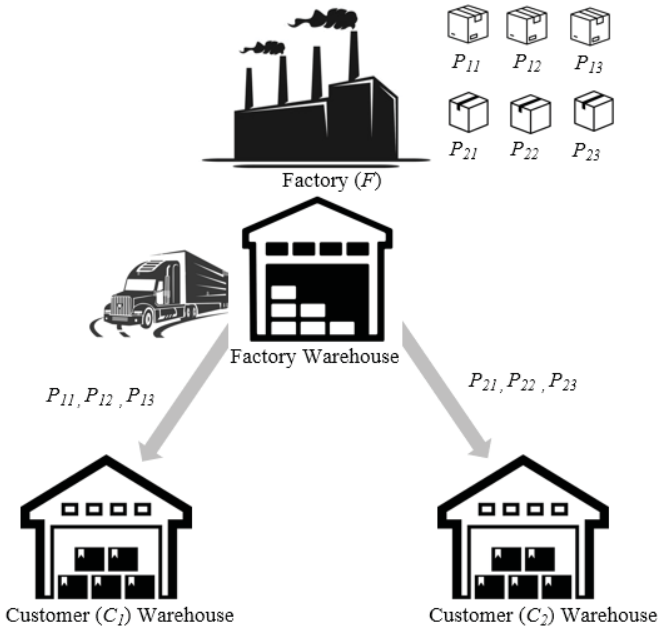


Fig. 1 Schematic diagram of the supply chain network with a single factory (F) and two customers C_1 and C_2, where six products P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, P_{23} are produced and shipped using separate storage and transports (for simplicity we show only a single transport and a single warehouse)

B. Modeling Supply Chain as Markov Decision Process

Markov Decision Process or MDP's are a classical formalization of sequential decision-making problems, where *actions* influence not just immediate *rewards*, but also subsequent situations, or *states*, and the future rewards [14]. In this section, we discuss how the supply chain optimization problem of the network shown in Fig. 1 can be formulated as a finite MDP. In a finite MDP, the sets of states, actions, and rewards all have a finite number of elements.

At any given time, t, the state space s_t comprises of product stock levels at the factory and customer warehouses. Thus,

$$s_t := \Big[W^t_{11,F}, W^t_{12,F}, W^t_{13,F}, W^t_{21,F}, W^t_{22,F}, W^t_{23,F}, W^t_{11,C_1}, W^t_{12,C_1}, W^t_{13,C_1}, W^t_{21,C_2}, W^t_{22,C_2}, W^t_{23,C_2}, \Big], \tag{1}$$

where, $W^t_{ij,F}$ denotes the stock level of product P_{ij} at the factory warehouse and W^t_{ij,C_1} denotes the stock level of product P_{ij} at customer C_1 warehouse at time t.

Actions comprise of daily product quantities shipped from factory to the customer warehouses. Thus, at any time t, action space a_t is:

$$a_t := \Big[Q^t_{11}, Q^t_{12}, Q^t_{13}, Q^t_{21}, Q^t_{22}, Q^t_{23}\Big], \tag{2}$$

where, Q^t_{ij} denotes quantity of P_{ij} shipped from the factory to the customer warehouses at time t. Based on (1) and (2), state transitions equations to the next state s_{t+1} can be written as:

$$\begin{aligned} s_{t+1} := \Big[&W^t_{11,F} + S^t_{11} - Q^t_{11}, ... \\ &W^t_{23,F} + S^t_{23} - Q^t_{23}, ... \\ &\max\{W^t_{11,C_1} + Q^t_{11} - D^t_{11}, 0\}, ... \\ &\max\{W^t_{23,C_2} + Q^t_{23} - D^t_{23}, 0\}\Big], \end{aligned} \tag{3}$$

where, S^t_{ij} and D^t_{ij} denote supply quantity and demand respectively for product P_{ij} at time t. Note that we limit the minimum value of the state variable representing customer warehouse storage levels to 0 - it is not feasible for warehouses to meet demands higher than the existing stock levels.

The one-step reward function, r_t, represents the profit gained during each time period t. The profit can be calculated as the total revenue minus the total cost for each time period. The total revenue is calculated by summing the product of the demand, D^t_{ij}, and the sales price, p_{ij}, for each product P_{ij}. Thus, total revenue, R^t, at time t is given by:

$$R^t = \sum_{i=1}^{2}\sum_{j=1}^{3} D^t_{ij} \cdot p_{ij} \tag{4}$$

The total cost is calculated by summing the production cost, C_{prod}, storage cost, C_{stor}, and transportation cost, C_{trans}, for each product. Thus, total cost, C^t_{total}, at time t is given by:

$$C^t_{\text{total}} = \sum_{i=1}^{2}\sum_{j=1}^{3}(C_{\text{prod},P_{ij}} + C_{\text{stor},P_{ij}} + C_{\text{trans},P_{ij}}), \tag{5}$$

where, $C_{\text{prod},P_{ij}}$ denotes production cost for product P_{ij} and so on. The reward function is then calculated as the difference between the total revenue (given by (4)) and total cost (given by (5)). Thus,

$$r_t = R^t - C^t_{\text{total}} \tag{6}$$

By maximizing the reward function over time, the supply chain can maximize its annual cumulative profit over an *episode* of a single year.

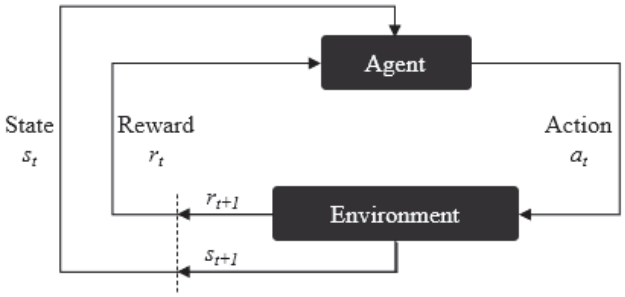


Fig. 2 Agent-environment interaction in an MDP: at time t and state s_t, the agent takes action a_t based on policy $\pi(a_t|s_t)$, receives reward r_t, and transitions to new state s_{t+1}; goal is to maximize long-term expected cumulative reward

IV. Methodology

In this section, we discuss basic RL and DRL concepts, and present experimental details of our study.

A. Reinforcement Learning Concepts

Fig. 2 depicts the fundamental learning mechanism in RL. At time step t, the agent takes an action a_t and the environment transitions from state s_t to state s_{t+1}, resulting in an immediate reward r_t. The agent then uses the state information s_{t+1} and the immediate reward r_t to select the next action a_{t+1}, and the cycle continues.

The optimal policy, denoted by $\pi(a_t = a|s_t = s)$, is a mapping from states to actions that maximizes the value function $v_\pi(s)$, which represents the long-term expected sum of future rewards:

$$v_\pi(s) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty}\gamma^t r_t|s_t = s\Big], \tag{7}$$

where $\mathbb{E}[.]$ denotes the expected value of a random variable given that the agent follows policy π, and t is the time step. Here, the discount factor $\gamma \in [0, 1]$ determines the importance of future rewards, with $\gamma = 0$ corresponding to a greedy agent that only cares about the reward at the next time step, and $\gamma = 1$ corresponding to an agent that values all future rewards equally. It is important to note that each value function $v_\pi(s)$ is tied to a specific policy π, which dictates the agent's future trajectory through the space of states.

RL algorithms can be thought of as iterative updates to a policy that improve the associated value function for all states. If the agent properly adjusts the policy at each iteration, the policy will continue to improve, resulting in larger and larger values for the value function at any given state. It is natural to wonder whether the value function ever reaches its optimal value, $v_*(s)$. Bellman's optimality equation, a necessary condition in optimal control theory, provides an answer to this question:

$$v_*(s) = max\sum_{s',r} p\Big(s', r|s, a\Big)\Big[r + \gamma v_*(s')\Big] \tag{8}$$

Here, the transition probability $p\Big(s', r|s, a\Big)$ calculates the probability that the environment will transition to state s' with

reward r given the current state s and action a. Bellman's optimality equation gives a set of nonlinear equations that, in theory, can be solved directly to produce the optimal value function $v_*(s)$ for a discrete set of actions and states. In practice, however, these equations are often not directly solvable due to either a lack of knowledge about the environment's transitions or the state space being too large to allow for a reasonable solution. All RL algorithms are therefore approximate solutions to Bellman's optimality equation, and they address these limitations in different ways.

For continuous states and actions, the state and action spaces can be enormous, making it impossible to derive exact solutions for the optimal value function. In these cases, function approximation methods, such as neural networks, are often used. Specifically, we can parameterize the approximate value function with parameters θ as follows:

$$v(s) \cong v_\theta(s) \tag{9}$$

Goal of the RL method in this case is to estimate θ that produce a value function that is as close to the actual value function as possible, for example, in terms of reducing the mean squared error (MSE):

$$L(\theta) = \frac{1}{2} \sum_s \Big[v_\theta(s) - v_*(s) \Big]^2 \tag{10}$$

B. Agent Training Experiments

We provide the experimentation details for training RL agents in this section.

a) Base Agents: In our initial experimentation, we utilize RL methodology to train agents for a scenario in which the demand for all six products, P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, and P_{23}, remains constant over the course of the year. We accomplish this by training and evaluating the agents using constant demand values, D_{ij}, for each product P_{ij}. We refer to the agents trained in this manner as *base agents* and it serves two primary purposes: *(i)* it provides a basis for comparison between the RL agents and a mathematical optimization benchmark, and *(ii)* it generates a set of base agents that can be compared to those specifically trained for handling sharp demand uncertainties.

We use a PPO algorithm for training agents throughout the entirety of the research [15], [16]. We use PPO implementation of Python library Stable Baselines and its default hyperparameter settings are employed [17]. Typically, we trained the agents for roughly 1.5×10^6 to 2×10^6 environment iterations, allowing enough time for a policy convergence. Since, we are provided the upper and lower limits for state and action variables (as given by (1) and (2) respectively), we normalize their values. Specifically, we normalize state space between [0, 1] and action space between [-1, 1] for assistance in policy training.

In RL, training a single agent on a particular task or environment can be highly stochastic and may not guarantee the optimal solution [18]. This is true as the optimization process involves many random variables, such as the initial state and reward distributions, which can result in different outcomes for the same algorithm and hyperparameters. Therefore, to account for the inherent variability of the method, we train 20 independent base agents. Training multiple agents also provides a basis for statistical analysis, allowing for the calculation of means, medians and distribution spreads of key metrics, which helps in drawing more reliable conclusions about the effectiveness of the RL algorithm.

A common challenge that arises in employing RL for constrained environments is dealing with *invalid actions* - during training, an action is taken which might result in the value of a state variable to lie outside of its prescribed bounds [21]-[23]. In our problem, this might lead to warehouse storages falling lower than the permissible lower limit or increasing beyond the maximum capacity during the state transition step (given by (3)). We use a *penalization* approach to de-incentivize agents from taking invalid actions by tailoring suitable penalty function. At each iteration of the environment, a penalty value is calculated based on the storage levels of products in factory and customer warehouses and is subtracted from the reward scalar.

b) Adaptive Agents: Generally, supply chains have at least some sort of variability in demand profiles. As such, it is expected that agents trained using fixed demand profiles may exhibit poor generalization performance when applied to scenarios characterized by stochasticity in product demands. To develop RL agents capable of learning generalizable policies for adapting to sharp single-day demand fluctuations, we devise a certain training procedure. Please note that to protect the confidentiality of the training protocol, its specific details have been obfuscated and we only provide a restricted version below.

Procedure – Strategy to design adaptive RL agents capable of handling product demand stochasticity.

Steps :

1: Add normally distributed random noise to the product demands with zero mean and a standard deviation of 1% - 2% of D_{ij}.
2: Sets of demand curves are generated, with each set having a certain % of the total days in a year where the demand for each product, D^t_{ij}, is increased by 0.50*D^t_{ij} (Fig. 3). For evaluation purposes, four sets of demand curves are generated with $\delta_1\%$, $\delta_2\%$, $\delta_3\%$, and $\delta_4\%$ of days featuring sharp demand spikes. Here $\delta_4\% > \delta_3\% > \delta_2\% > \delta_1\%$.
3: *Training:* In the training phase, to diversify the learning experience of the agents, demand curves are utilized based on a design scheme, and 20 agents are trained independently.
4: *Evaluation:* 20 trained agents are tested on the test demand curves and the performance is quantified.

We refer to the agents trained in the described manner as *Adaptive Agents*. In addition, we design two analogous scenarios by making minute changes in **Step 2** of the procedure above: *(i)* instead of increasing, the product demands are reduced for a certain percentage of days by 0.50*D^t_{ij}, and *(ii)* product demands can randomly increase or

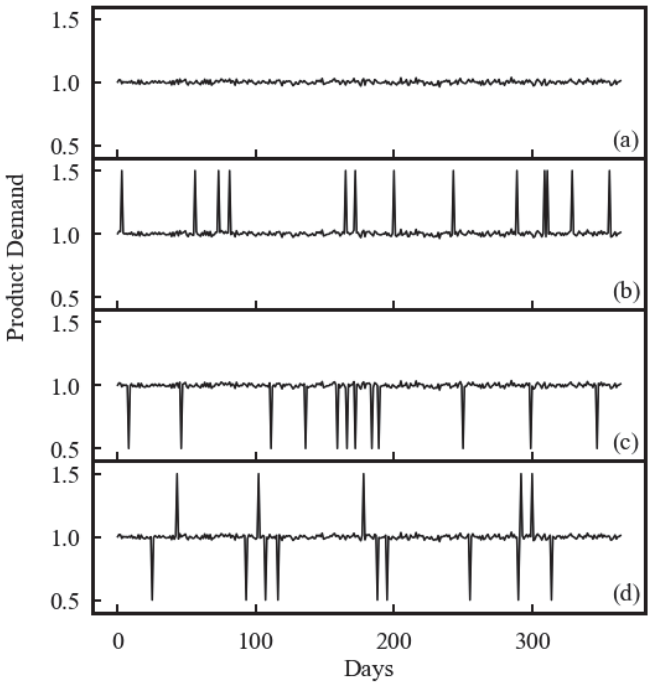


Fig. 3 (a) Product demand D_{11} (on a normalized/dummy scale) with zero mean and a standard deviation of 1% - 2% of D_{ij}; (b) Example demand profile of Scenario I, with a certain % of days with an increased demand by 0.50*D_{ij}; (c) Example demand profile of Scenario II, with a decreased demand by 0.50*D_{ij}; (d) Example demand profile of Scenario III, having days with random increased and decreased demand by 0.50*D_{ij}

decrease by 0.50*D^t_{ij}. Resulting perturbed product demand, D_{11} is shown for these three scenarios in Figs. 3 (b), (c) and (d) respectively for a certain % of *exceptional* days in a year. Fig. 3 (a) specifically demonstrates the effects of including random noise, as mentioned previously in **Step 1** of the aforementioned procedure. Note that values are presented on a normalized scale. Throughout, the rest of this paper, these three types of scenarios are referred as *Scenario I*, *Scenario II* and *Scenario III* respectively:

TABLE I
DIFFERENT DEMAND SCENARIOS STUDIED IN THIS RESEARCH

Scenario	Demand disruptions type
I	Sharp Increase
II	Sharp Decrease
III	Volatile Demand

Please note that although we do not provided the actual values of $\delta_1\%$, $\delta_2\%$, $\delta_3\%$, and $\delta_4\%$, they are the same across the three scenario types.

V. RESULTS

In this section, we discuss the findings from our RL-based approach. We first compare base agents against a Linear Programming benchmark for fixed demand scenario and later discuss the performance of adaptive RL agents.

A. Base Agent comparison with Linear Programming Benchmark

We train 20 independent base agents, following the procedure outlined in Section IV B. For comparison purposes, we formulate the supply chain optimization problem for constant demand as a Linear Programming (LP) problem [20]. We deliberately compare the performance of RL against a global optimization benchmark which is considerably more challenging than fixed-policy benchmark. However, it should be noted that during our in-house experimentation, RL demonstrated its ability to easily surpass fixed-policy benchmarks.

We model continuous variables for product stock levels in the warehouses and product amounts shipped to customers, while capturing product influx and outflux at the warehouses with *material balance* constraints. The objective function aims to maximize the net annual profit. We use Pyomo [19] to model the problem and GLPK, an open-source mathematical optimization solver to find the optimal solution. The optimal shipping Q_{ij} of product P_{ij} for all the six products are shown in Fig. 4 (a). Fig. 4 (b) shows sample product shipping plan from RL approach. We observe a significant difference in product shipping patterns between the two approaches, with the shipping pattern from RL exhibiting far higher noise. The LP method generates a net profit of 18.7 million USD for the objective function, whereas the highest profit obtained from RL was 18.4 million USD, which is still in close proximity ($< 3\%$).

While the RL method did not outperform the LP method, it was able to achieve a performance within 3% of the latter. This result is consistent with the observation that RL may not always be the most efficient or effective approach for global optimization problems. However, we note that the strength of RL lies in its adaptability, which enables it to learn and adjust to changing environments and objectives. As such, RL may be a more suitable approach for optimization problems that require adaptability and flexibility, rather than global optimization. We present supporting evidence for this assertion in the next section.

B. Adaptive Agents Comparison with Base Agents

Based on the methodology described in Section IV B, we train and evaluate RL agents for Scenario I, II, III and compare the performance of adaptive agents against base agents and quantify their robustness. In order to ensure reliable statistical analysis, we again train 20 independent agents for each of the presented scenarios. Specifically, we compare performance over two metrics: *(i)* annual cumulative profit generated, and *(ii)* net constraint violation. While the former metric is a key parameter of interest, the latter indicates the validity of learnt optimal policy by the agents.

We quantify constraint violation by measuring the deviation of the variables denoting warehouse product stock levels from the predefined minimum and maximum storage levels. For a single RL agent, we first obtain annual timeseries data for all twelve product stock levels. For a given variable value, we then calculate the percentage by which the variable violates the

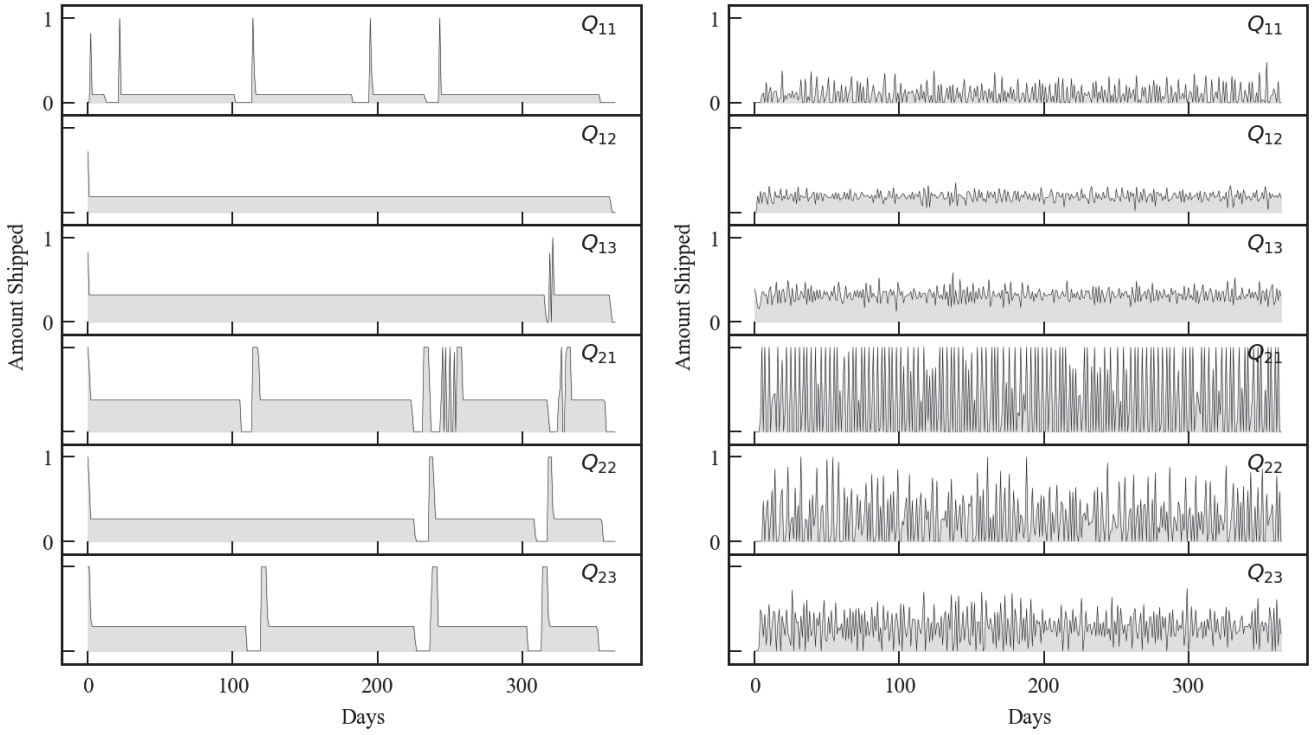


Fig. 4 Optimal amount of products shipped Q_{11}, Q_{12}, Q_{13}, Q_{21}, Q_{22}, and Q_{23} based on (a) linear programming approach, and (b) RL approach

minimum or maximum storage level constraint and accumulate this as a sum. We repeat the above procedure for the ensemble of 20 RL agents and maintain a running sum of percentages. Finally, we normalize the sum of these percentage values by 20 $\times$365$\times$12 to get an estimate of total constraint violation.

We show the results in Fig. 5, where (a), (b) and (c) show boxplots for cumulative profits for scenarios I, II and III respectively, and the constraint violation is shown in Figs. 5 (d), (e) and (f) respectively. Boxplots are generated with a sample size of roughly 1000. We draw the following conclusions:

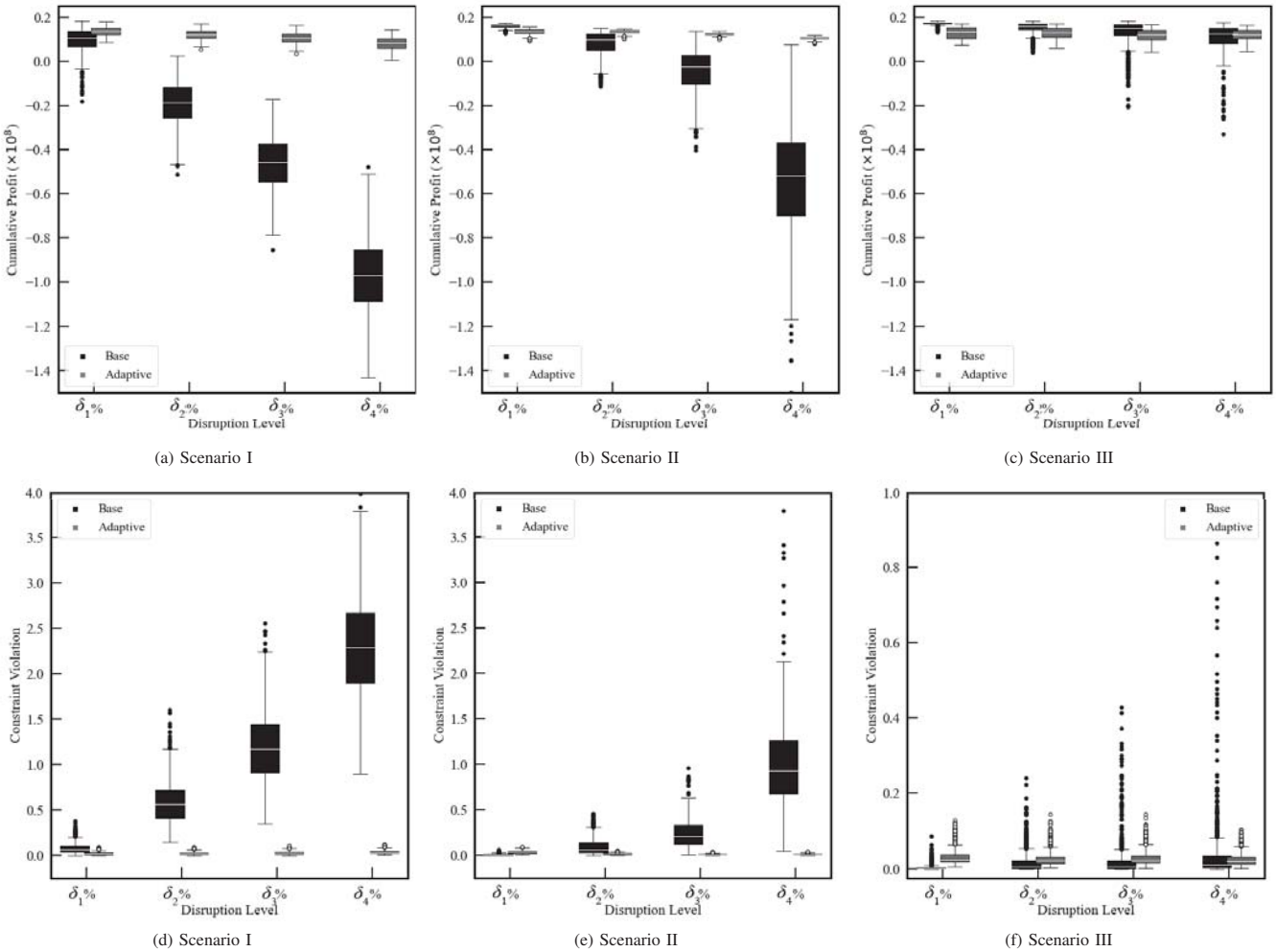
a) Scenario I: For $\delta_1\%$ peak demand days, base agents and adaptive agents generate similar median cumulative profit, though comparatively base agents display significant distribution spread. The difference, however appears increasingly pronounced for $\delta_2\%$, $\delta_3\%$ and $\delta_4\%$ peak demand days where adaptive agents significantly outperform base agents which actually fail to generate any profit. Negative cumulative profit indicates unmet demand. A similar trend appears to exist for constraint violation as well, where base agents display sharp increase in constraint violation with increased percentage of peak demand days. Adaptive agents, on the other hand perform extremely well and consistently maintain very little constraint violation.

b) Scenario II: We observe almost identical trends as Scenario I - adaptive agents largely outperform base agents. Despite this, there is a small variation: performance deterioration for higher % of anomalous demand days is comparatively less than as observed in Scenario I. The same applies for constraint violation as well. This is consistent with the fact that increased demand is more likely to add to penalty incurred and reduced profits.

c) Scenario III: Interestingly, we observe that base agents perform similar to adaptive agents to a large extent for this scenario. In fact, based on median cumulative profit, base agents slightly outperform adaptive agents for $\delta_1\%$, $\delta_2\%$ and $\delta_3\%$ of peak or low demand days. For the last case agents appears to be performing slightly better. Simultaneously, for extreme cases, base agents display significant dispersion of outlier data towards one side of the median. The observed phenomenon can be attributed to the possibility that the training scheme for the RL agents for this scenario is excessively challenging, leading to an inability for the agents to extract meaningful information from the abundant variations presented in the demand curves during training. As a result, these agents exhibit similar performance to the base agents and do not appear to surpass their performance levels by a significant margin.

VI. Conclusion

In conclusion, our research explores the use of Reinforcement Learning (RL) for supply chain optimization problems and contributes to the existing knowledge within this domain. Specifically, we study scenarios with no stochasticity in product demands and with varying levels of sharp, single-day variations in product demand. We used PPO algorithm to train RL agents and use a specific penalization approach for handling system constraints. We provide experimental details for training agents that demonstrate capabilities of devising adaptive policies for



(a) Scenario I (b) Scenario II (c) Scenario III

(d) Scenario I (e) Scenario II (f) Scenario III

Fig. 5 Evaluation of base and adaptive agents on test demand curves with $\delta_1\%$, $\delta_2\%$, $\delta_3\%$, and $\delta_4\%$ anomalous demand days (scenarios I, II, III): (a)-(c) boxplots for cumulative profit, and (d)-(f) boxplots for constraint violation for scenarios I, II, III. Analysis based on 1000 samples, with base agents shown in black and adaptive agents in gray (positioned right relatively)

dealing with product uncertainty. Results indicate that the RL approach performs comparably with linear programming benchmark and provides better adaptability to stochasticity in product demands. Overall, the results suggest that RL can be a viable alternative for supply chain optimization problems, particularly in scenarios with stochasticity in product demands. Extensions of our work include designing other training schemes for adaptive RL agents. A future study can involve exploring other types of variations in the product demand, e.g., continuous multi-day fluctuations or seasonal demand patterns.

REFERENCES

[1] T. de Kok, C. Grob, M. Laumanns, S. Minner, J. Rambau, and K. Schade, "A typology and literature review on stochastic multi-echelon inventory models," European Journal of Operational Research, vol. 269, no. 3, pp. 955–983, 2018.

[2] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro, "A stochastic programming approach for supply chain network design under uncertainty," European Journal of Operational Research, vol. 167, no. 1, pp. 96–115, 2005.

[3] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," nature, vol. 529, no. 7587, pp. 484–489, 2016.

[4] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," arXiv preprint arXiv:1704.02532, 2017.

[5] X.-Y. Liu, Z. Xiong, S. Zhong, H. Yang, and A. Walid, "Practical deep reinforcement learning approach for stock trading," arXiv preprint arXiv:1811.07522, 2018.

[6] F. Stranieri and F. Stella, "A deep reinforcement learning approach to supply chain inventory management," arXiv: 2204.09603, 2022.

[7] E. E. Kosasih and A. Brintrup, "Reinforcement Learning Provides a Flexible Approach for Realistic Supply Chain Safety Stock Optimisation," IFAC-PapersOnLine, vol. 55, no. 10, pp. 1539–1544, 2022.

[8] S. Kumabe, S. Shiroshita, T. Hayashi, and S. Maruyama, "Learning General Inventory Management Policy for Large Supply Chain Network," arXiv preprint arXiv:2204.13378, 2022.

[9] J. César Alves and G. Robson Mateus, "Multi-echelon Supply Chains with Uncertain Seasonal Demands and Lead Times Using Deep Reinforcement Learning," arXiv e-prints, p. arXiv-2201, 2022.

[10] L. Kemmer, H. von Kleist, D. de Rochebouët, N. Tziortziotis, and J. Read, "Reinforcement learning for supply chain optimization," in European Workshop on Reinforcement Learning, 2018, vol. 14, no. 10.

[11] E. E. Kosasih and A. Brintrup, "Reinforcement Learning Provides a Flexible Approach for Realistic Supply Chain Safety Stock Optimisation," IFAC-PapersOnLine, vol. 55, no. 10, pp. 1539–1544, 2022.

[12] H. Meisheri et al., "Using reinforcement learning for a large variable-dimensional inventory management problem," Adaptive Learning Agents Workshop, AAMAS, 2020.
[13] Z. Peng, Y. Zhang, Y. Feng, T. Zhang, Z. Wu, and H. Su, "Deep reinforcement learning approach for capacitated supply chain optimization under demand uncertainty," in 2019 Chinese Automation Congress (CAC), 2019, pp. 3512–3517.
[14] A. L. Strehl, L. Li, and M. L. Littman, "Reinforcement Learning in Finite MDPs: PAC Analysis.," Journal of Machine Learning Research, vol. 10, no. 11, 2009.
[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.
[16] Y. Wang, H. He, and X. Tan, "Truly proximal policy optimization," in Uncertainty in Artificial Intelligence, 2020, pp. 113–122.
[17] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," The Journal of Machine Learning Research, vol. 22, no. 1, pp. 12348–12355, 2021.
[18] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in Proceedings of the AAAI conference on artificial intelligence, 2018, vol. 32, no. 1.
[19] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in Python," Mathematical Programming Computation, vol. 3, pp. 219–260, 2011.
[20] G. B. Dantzig, "Linear programming," Operations research, vol. 50, no. 1, pp. 42–47, 2002.
[21] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," Computers & Chemical Engineering, vol. 139, p. 106886, 2020.
[22] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," arXiv preprint arXiv:1801.08757, 2018.
[23] E. Pan, P. Petsagkourakis, M. Mowbray, D. Zhang, and E. A. del Rio-Chanona, "Constrained model-free reinforcement learning for process optimization," Computers & Chemical Engineering, vol. 154, p. 107462, 2021.

Nitin Singh is a professional in operations research with over 4 years of experience in numerical optimization, statistics and data science. He received his Ph.D. degree in computational physics from Leiden University in 2019. Currently, he works as an Operations Research Analyst at Shell plc, Bangalore, where he has led and delivered several projects related to supply chain, logistics, power optimization, and process engineering. His expertise lies in supply chain, deep reinforcement learning, machine learning and numerical optimization.

Meng Ling is an accomplished AI Researcher with expertise in computer vision, natural language processing, and reinforcement learning, as well as a seasoned data scientist in the fields of environmental science, energy, and public health. Currently employed at Shell plc in Houston, he holds a Ph.D. degree with expertise in engineering design, numerical modelling, data analytics, and statistics. His current research focuses on utilizing deep reinforcement learning and graph neural networks to optimize supply chains and other industrial applications.

Talha Ahmed is a 4^{th} year Ph.D. student at the University of Tennessee, Knoxville, USA majoring in electrical engineering with a minor in Computer Science. Prior to his Ph.D., he received his bachelor of science degree from National University of Sciences and Technology in 2017 from Islamabad, Pakistan. His research interests lie in the areas of deep learning and reinforcement learning along with expertise in applying them to nonlinear dynamical systems.

Tianxia(Tina) Zhao is the Industrial Artificial Intelligence Team Lead working at Shell technology center in Houston. She has worked and led many data science projects using data analytics, machine learning, deep learning, reinforcement learning, NLP, and optimization. She is highly experienced in many aspects of the oil and gas industry including commercial insight, customer centricity, trading, asset management/predictive maintenance, petrochemicals, new energy system including power and Hydrogen, and operation optimization. She has a strong research background with a Ph.D. degree in electrical engineering from University of Houston.

Reinier van de Pol has a masters degree in applied mathematics from University of Twente in the Netherlands. He joined Shell in 1992 and held different roles in Upstream, Downstream and Integrated Gas with increasing focus on supply chain logistics and optimization Currently, he works at Shell plc, Amsterdam and is the Team Lead of the Operations Research team within the Decision Science group in Projects & Technology.