A Deep Reinforcement Learning Approach to Supply Chain Inventory Management

Francesco Stranieri[†‡] and Fabio Stella[†]

[†]Department of Informatics, Systems, and Communication, University of Milano-Bicocca
[‡]Department of Control and Computer Engineering, Polytechnic University of Turin

Introduction

Supply chain inventory management (SCIM) is a *sequential decision-making problem* consisting of determining the optimal quantity of products to produce at the factory and to ship to different distribution warehouses over a given time horizon. Deep reinforcement learning (DRL) algorithms are rarely applied to the SCIM field, although they can be used to develop near-optimal policies that are difficult, or impossible at worst, to achieve using traditional mathematical methods [1]. Moreover, when effectively applied, they suffer several *limitations* [2].

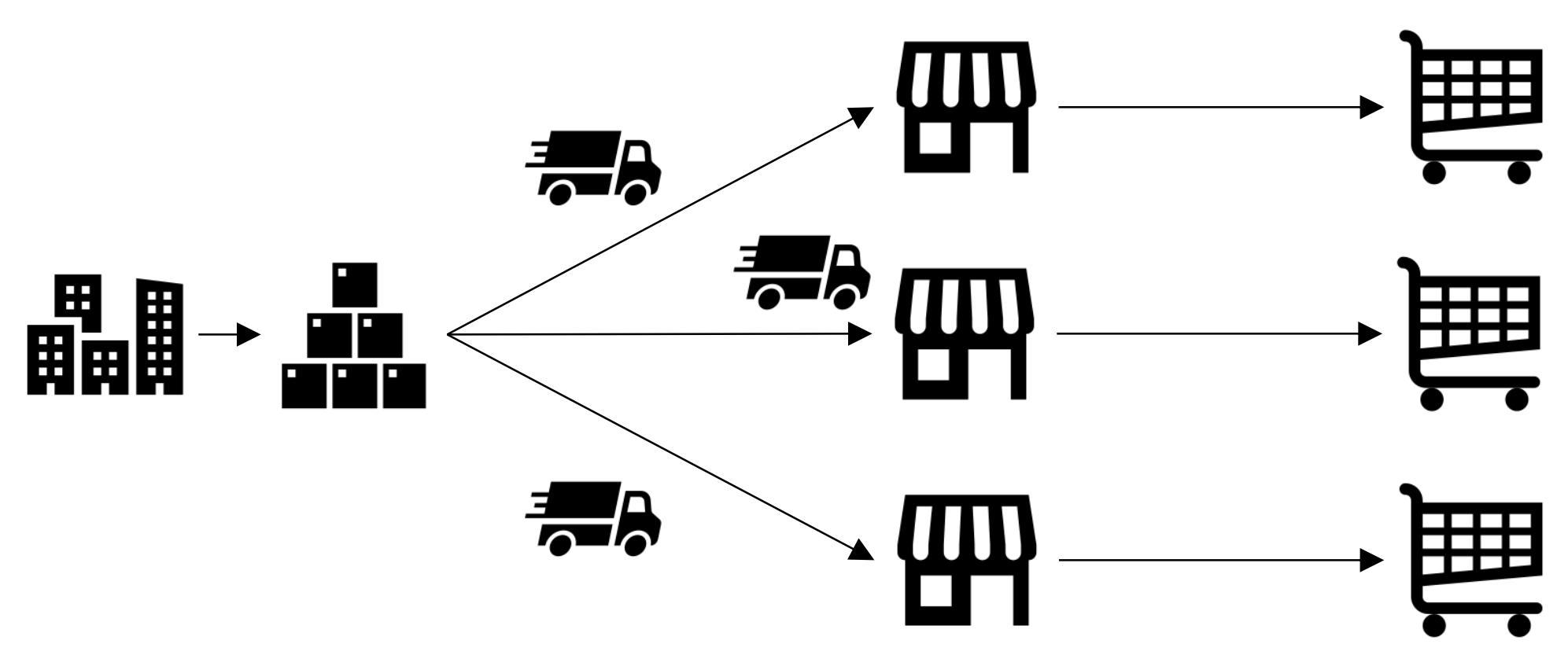

Fig. 1: A divergent two-echelon supply chain consisting of a factory and its warehouse (first echelon), plus three distribution warehouses (second echelon). Shopping carts represent customers' demands.

In our research, a mathematical formulation of the SCIM environment is given, which includes a *factory* that can produce various *product types*, a *factory warehouse*, and a certain number of *distribution warehouses*. To compare DRL algorithms performances, a rich set of *numerical experiments* on synthetically generated data have been designed and performed under three different scenarios: *one product type one distribution warehouse* (1P1W), *one product type three distribution warehouses* (1P3W), and *two product types two distribution warehouses* (2P2W). Each scenario involves different warehouse capacities, margin of return, demands (both in terms of maximum value and uncertainty), and costs as well as different values of hyperparameters associated with DRL algorithms.

Problem Definition

For each product type i, each warehouse j has a maximum capacity of $c_{i,j}$ ($\sum_{i=0}^{I} c_{i,j} = c_j$), a storage cost of $z^S_{i,j}$ per unit, and a stock level at time t equal to $q_{i,j,t}$. The DRL algorithms determine for each product type i and time step t how many units to produce, i.e., $a_{i,j,t}$ with $j = 0$, considering a fixed production cost of $z_{i,0}$ per unit, and how many units ship to each distribution warehouse, i.e., $a_{i,j,t}$ with $1 \leq j \leq J$, assuming a transportation cost of $z^T_{i,j}$ per unit. Each unit of product type i is sold to customers at sale price p_i, while the (stochastic) demand at distribution warehouse j for time step t is equivalent to $d_{i,j,t}$ units.

Variable	Explanation	Variable	Explanation
I	Number of Product Types	$q_{i,j,t}$	Stock Level (units)
J	Number of Warehouses	$c_{i,j}$	Storage Capacity (units)
T	Episode Length	$z^S_{i,j}$	Storage Cost (per unit)
$a_{i,j,t}$	Production and Shipping Level (units)	z^P_i	Penalty Coefficient
$z_{i,0}$	Production Cost (per unit)	p_i	Sale Price (per unit)
$z^T_{i,j}$	Transportation Cost (per unit)	$d_{i,j,t}$	Demand (units)

Tab. 1: The considered SCIM variables with relative explanation (and units of measure). All these variables are integrated and editable within our open-source library (available on `https://github.com/frenkowski/SCIMAI-Gym`).

Products are non-perishable and provided in discrete quantities. If an order for a certain time step exceeds the corresponding stock level, a penalty cost is applied (obtained by multiplying the penalty coefficient z^P_i with p_i). Unsatisfied orders are also maintained over time and are designed as a negative stock level (this corresponds to *backordering*).

MDP Formulation

For the *state vector*, we include all current stock levels for each warehouse and product type, plus the last $\tau(=5)$ demand values:

$$\mathrm{s}_t = (q_{0,0,t}, \ldots, q_{I,J,t}, d_{t-\tau}, \ldots, d_{t-1}),$$

where $d_{t-1} = (d_{0,1,t-1}, \ldots, d_{I,J,t-1})$.
Concerning the *action vector*, we implement a *continuous action space* (i.e., the neural network generates the action value directly):

$$\mathrm{a}_t = (a_{0,0,t}, \ldots, a_{I,J,t}).$$

Our implementation provides an *independent upper bound* for each action value; for each distribution warehouse, it corresponds to its maximum capacity with respect to each product type ($0 \leq a_{i,j,t} \leq c_{i,j}$), while for the factory to the sum of all warehouses' capacities with regard to each product type ($0 \leq a_{i,0,t} \leq \sum_{j=0}^{J} c_{i,j}$).
To simulate a *seasonal behavior*, we represent the *demand* as a co-sinusoidal function with a stochastic component:

$$d_{i,j,t} = \left\lfloor \frac{d_{max_i}}{2}\left(1 + \cos\left(\frac{4\pi(2ij+t)}{T}\right)\right) + \mathcal{U}\left(0, d_{var_i}\right) \right\rfloor,$$

where $\lfloor\cdot\rfloor$ is the floor function, d_{max_i} is the maximum demand value for each product type, $\mathcal{U}$ is a random variable uniformly distributed on the support $(0, d_{var_i})$ representing the *uncertainty*, and T is the final time step of the episode.

The DRL algorithms' goal is to *maximize the supply chain profit*. Accordingly, we design the *reward function* for time step t as:

$$\begin{aligned}\mathrm{r}_t = &\sum_{j=1}^{J}\sum_{i=0}^{I} p_i \cdot d_{i,j,t} - \sum_{i=0}^{I} z_{i,0} \cdot a_{i,0,t} - \sum_{j=1}^{J}\sum_{i=0}^{I} z^T_{i,j} \cdot a_{i,j,t} \\ &- \sum_{j=0}^{J}\sum_{i=0}^{I} z^S_{i,j} \cdot \max(q_{i,j,t}, 0) + \sum_{j=0}^{J}\sum_{i=0}^{I} z^P_i \cdot p_i \cdot \min(q_{i,j,t}, 0).\end{aligned}$$

The first term represents revenues, the second one production costs, while the third one corresponds to transportation costs. The fourth term is the overall storage costs. The last term denotes the penalty costs.
Finally, we define the *state's updating rule* as follows:

$$\begin{aligned}s_{t+1} = \big(&\min[(q_{0,0,t} + a_{0,0,t} - \sum_{j=1}^{J} a_{0,j,t}), c_{0,0}], \cdots, \\ &\min\left[\left(q_{I,J,t} + a_{I,J,t} - d_{I,J,t}\right), c_{I,J}\right], d_{t+1-\tau}, \cdots, d_t\big).\end{aligned}$$

It is worth noting that the actual demand d_t will not be known until the next time step.

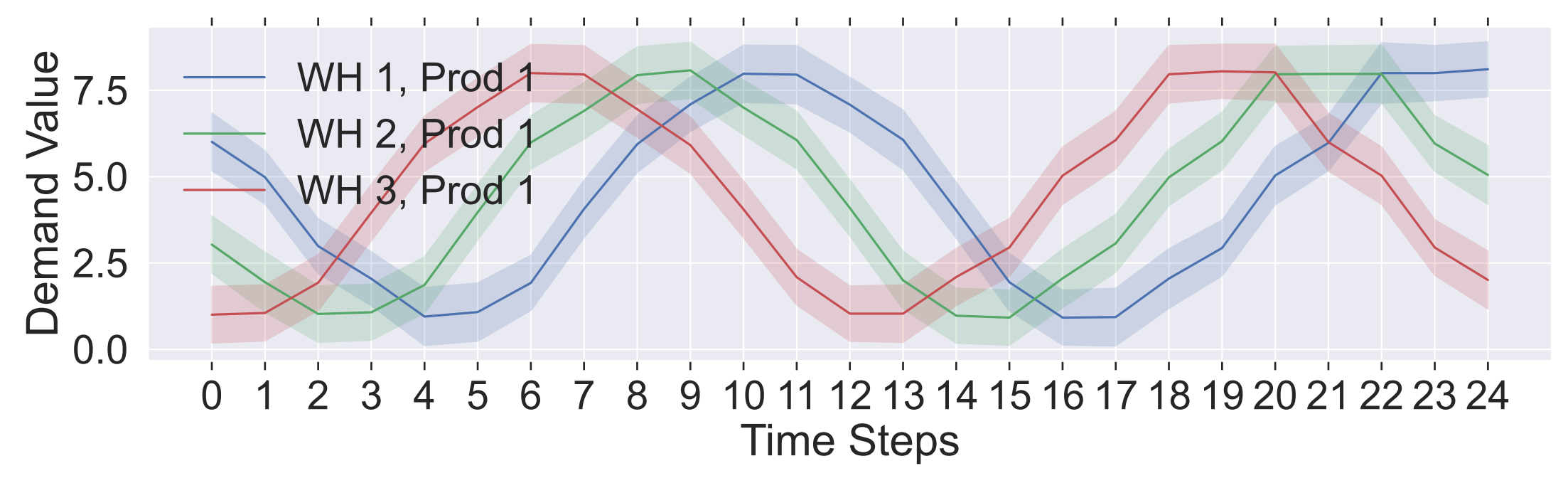

Fig. 2: An instance of the demand behavior considering the 1P3W scenario with $d_{max} = 7$ and $d_{var} = 2$.

Results

Performances achieved by state-of-the-art DRL algorithms (i.e., A3C, PPO, and VPG) are compared with a static reorder policy, i.e., an (s, Q)-policy whose optimal parameters have been set through a *data-driven* Bayesian optimization (BO) approach, and with an *oracle*, i.e., a baseline who knows the optimal action to take a priori. The results of numerical experiments demonstrate that the SCIM environment we propose is *effective* in representing states, actions, and rewards; indeed, DRL algorithms, and in particular PPO, have been able to learn nearly optimal policies in all three investigated scenarios.

	A3C	PPO	VPG	BO	Oracle
Exp 1	870 ± 67	1213 ± 68	885 ± 66	$\mathbf{1226 \pm 71}$	1474 ± 45
Exp 2	1066 ± 94	1163 ± 66	1100 ± 77	$\mathbf{1224 \pm 60}$	1289 ± 68
Exp 3	-36 ± 74	$\mathbf{195 \pm 43}$	12 ± 61	101 ± 50	345 ± 18
Exp 4	1317 ± 60	1600 ± 62	883 ± 95	$\mathbf{1633 \pm 39}$	2046 ± 37
Exp 5	736 ± 45	838 ± 58	789 ± 51	$\mathbf{870 \pm 67}$	966 ± 55

Tab. 2: Results covering the 1P1W scenario. BO and PPO achieve a near-optimal profit in the first experiment, where the demand is greater than the warehouses' capacities. All DRL algorithms obtain comparable results in the second and simpler experiment, while PPO tends to behave better in the third and more complex experiment. BO, PPO, and A3C obtain satisfactory profits in the fourth and more balanced experiment represented by a wider search space. In the fifth experiment, the uncertainty increases and penalties decrease, but all DRL algorithms achieve comparable and near-optimal results.

	A3C	PPO	VPG	BO	Oracle
Exp 1	1606 ± 139	$\mathbf{2319 \pm 122}$	803 ± 154	486 ± 330	3211 ± 60
Exp 2	2196 ± 104	$\mathbf{3461 \pm 120}$	2568 ± 112	3193 ± 101	3848 ± 95
Exp 3	-2142 ± 128	-4337 ± 216	-2638 ± 121	$\mathbf{-1682 \pm 196}$	772 ± 21
Exp 4	-561 ± 237	$\mathbf{2945 \pm 135}$	656 ± 140	1256 ± 170	4389 ± 64
Exp 5	1799 ± 306	$\mathbf{2353 \pm 131}$	1341 ± 79	2203 ± 152	2783 ± 91

Tab. 3: Results regarding the 1P3W scenario. In the first experiment, characterized by a high demand, BO performs worse than DRL algorithms. However, it achieves a nearly optimal profit in the second and simpler experiment. In the third and more challenging experiment, none of the algorithms obtains a profit greater than zero. PPO outperforms A3C and VPG in the fourth and more balanced experiment, characterized by an increased search space. Finally, BO and PPO achieve the best profits in the fifth experiment, where uncertainty and search space are increased but fewer penalties are considered.

	A3C	PPO	VPG	BO	Oracle
Exp 1	2227 ± 178	$\mathbf{2783 \pm 139}$	1585 ± 184	2086 ± 173	3787 ± 102
Exp 2	1751 ± 83	$\mathbf{2867 \pm 90}$	2329 ± 98	2246 ± 114	3488 ± 63
Exp 3	1414 ± 128	$\mathbf{2630 \pm 138}$	2434 ± 156	552 ± 268	3549 ± 103

Tab. 4: Results concerning the 2P2W scenario. The first experiment provides a balanced configuration, and PPO obtains a good profit, as it also does A3C, which overcomes BO. For the second experiment, penalties are increased, but PPO still achieves a nearly optimal result, and the same happens for VPG and BO. In the third experiment, with alternating storage costs, PPO, followed by VPG, continues to perform successfully, whereas BO seems to suffer the most.

References

[1] Robert N Boute et al. "Deep reinforcement learning for inventory control: A roadmap". In: *European Journal of Operational Research* (2021).

[2] Yimo Yan et al. "Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities". In: *Transportation Research Part E: Logistics and Transportation Review* 162 (2022), p. 102712.