A Reinforcement Learning Powered Digital Twin to Support Supply Chain Decisions

Guillaume Martin, Raphaël Oger

A Reinforcement Learning Powered Digital Twin to Support Supply Chain Decisions

Guillaume Martin
IMT Mines Albi
gmartin@mines-albi.fr

Raphaël Oger
IMT Mines Albi
raphael.oger@mines-albi.fr

Abstract

The complexity of making supply chain planning decisions is growing along with the Volatility, Uncertainty, Complexity and Ambiguity of supply chain environments. As a consequence, the complexity of designing adequate decision support systems is also increasing. New approaches emerged for supporting decisions, and digital twins is one of those. Concurrently, the artificial intelligence field is growing, including approaches such as reinforcement learning. This paper explores the potential of creating digital twins with reinforcement learning capabilities. It first proposes a framework for unifying digital twins and reinforcement learning into a single approach. It then illustrates how this framework is put into practice for making supply and delivery decisions within a drug supply chain use case. Finally, the results of the experiment are compared with results given by traditional approaches, showing the applicability of the proposed framework.

Keywords Digital Twin, Reinforcement Learning, Supply Chain Management, Deep Learning

1. Introduction

A supply chain can be defined as a "*network of connected and interdependent organizations mutually and co-operatively working together to control, manage and improve the flow of materials and information from suppliers to end users.*" [1] As such, they face several systemic cooperation challenges : a fragile equilibrium between cooperation and competition, opposed views between locally defined information sharing strategies and globally enforced ones, or the need for interopable communication (with interoperability defined as "*the ability of independent logistics and supply networks to mutually conduct operations and business with one another, in order to use the functionality of other networks, or to execute operations for others*" [2]).

In parallel with endogenous issues, a supply chain also faces external hurdles. The most common ones are fluctuating demands, either from errors in prediction or from sudden crisis situations, as the world witnessed during the pandemic [3]. Uncertainty also comes in the form of sudden disruptions in the chain, be they from key supplier bankruptcy or environmental issues [4].

In spite of this, the scientific community regularly shares new approaches to overcome the problems, by leveraging the power of the latest advances in communication and computing technology. Recent contributions include the use of RFID tagging of goods [5], GPS tracking of vehicles [6] or using the Internet of Things to enable communication between packages, sensors and systems as a whole [7]. Some of the contributions go further and describe systems using features from the physical and digital words to achieve a common goal. Cyber-Physical Systems, for example, use smart components such as storage units or machines, able to communicate in real time with other components through a common interoperable network [8]. Other advances use digital copies of the physical systems in order to monitor status and plan actions. These are called digital twins [9]. In these cases, a simulation and/or optimization engine is hooked up to the digital copy to simulate the impact of decisions.

A common feature of all these systems is that they are composed of agents. Each agent either has to communicate or make decisions based on the supply chain status. For the system to work correctly, these agents have to make intelligent decisions. Intelligence, in this case, may come in various forms: (i) decision-trees of rules based on practitioner experience, such as raising an alert on certain levels, (ii) artificial intelligence sub-systems with a specific task, for example with routing problems or (iii) human agents.

Such a division into agents is also the basis of a field called Reinforcement Learning (RL). RL uses agents in an environment that select actions (setting an reorder quantity, choosing between suppliers, ...) with the information of the state of the environment so as

URI: https://hdl.handle.net/10125/79620
978-0-9981331-5-7

to maximize an expected reward [10]. In RL, agents are defined as "*the learners and decision makers*" of the problem, while the environment is "*everything the agent interacts with*" [10]. RL has been evolving quickly in the last decade with the advent of deep neural networks and their ability to model complex decisions, leading to the creation of Deep Reinforcement Learning (DRL). A key feature of the RL field is to build and train intelligent agents not on rules but on how the environment reacts to actions. RL can consequently capture several degrees of complexity, depending on what is relevant for the task at hand, while avoiding some biases of the rule-based systems.

This paper proposes a framework of a supply chain digital twin where decisions are taken by a DRL agent. This helps bridge the gap between advances in computer science and supply chain control. This also serves as a template for systems where uncertainties are only known through data collection and cannot be accurately represented by common models. Section 2 reviews the recent related works on connected supply chain control and RL or DRL for supply chain decision-making. Section 3 details both our proposed framework, how to train it offline and how to use it online. Section 5 shows numerical results of training and using such an agent on an illustrative supply chain. Lastly, Section 6 concludes on the next steps needed to consolidate our contribution.

2. Related works

This review consists of three subsections. The first is dedicated to existing works on connected supply chains. It includes works on the physical internet, digital twins for better supply chain management. The second is dedicated to Reinforcement Learning and its ability to help supply chain management achieve better performance. The third concludes with our research question paper.

2.1. Connected supply chain management

Physical internet, as proposed by [11], is defined as a network of physical and digital elements sharing common interfaces. For a broad picture of applications and research themes, [12] recently published a thorough survey on the topic. As for more specific works, [13] show how to implement physical internet specific routing. [14] propose physical internet specific containers which are active in their own routing and [15] describe a physical internet city logistics including physical internet picking waves and distribution. Applied to supply chain management, physical internet approaches make heavy use of decentralizing: common rules are set up to simplify the decision making between each member (human or otherwise) of the decision processes. For example, [16] describe a decentralized inventory management use case.

Besides physical internet implementations, some authors recommend using digital twins of supply chains to simulate the effects of their decisions. A digital twin is defined in [9] as a copy of the physical supply chain with data coming from "*actual transportation, inventory, demand, and capacity*". A digital twin can either be "offline" and represent a single point in time from where we investigate possible decisions, on "online" and "used for planning and real-time control decisions" [9]. [17] surveyed the literature on digital twins for the last 20 year and summarized their definitions, key features and applications. The surveyed papers close to our work focus on digital twins as "*integrated systems, acting as virtual clones with data connections to the real system and some simulation or prediction capabilities*". The authors also note a sharp increase in papers with manufacturing applications of digital twins.

[18] use a digital twin in a large cyber-physical system applied to a manufacturing case. The authors describe the different layers needed to tie the physical and digital levels together in a shared system. [19] goes over a common framework for using digital twins in an Industry 4.0 context: a metadata model to ensure digital interoperability, a simulation framework to model the system replica and a communication layer to bound physical and digital worlds together in real-time. [20] propose a digital twin framework for crisis management in a city logistics context. They split their framework into four major components: data collection, data integration, multi-agent decision making and dynamic network analysis. Lastly, [21] define an on-demand shared digital twin for supply chain management. They raise several key problems: (i) data sovereignty issues may happen along the chains, especially global ones and (ii) data formats should be as interoperable as possible, maybe even language and model agnostic.

2.2. Reinforcement learning for supply chain decision-making

Reinforcement learning (RL) and Deep reinforcement learning (DRL) have been successfully applied to various sub-areas of supply chain decision-making. The contributions mainly differ by two criteria: the algorithm used and the number of agents involved. For brevity's sake, we do not describe the algorithms in this paper however. [22], for example, use a single agent with complete information to dispatch production in a cloud manufacturing environment. [23]

propose a way to solve the joint distribution problem: which items to select for which types of deliveries. In the logistics area, [24] show how to assign vehicles to platoons of trucks and where to send the platoons. [25] compare different ways to decide reordering quantities in an multi stage inventory management problem. The authors show that DRL policies are more stable than human-like ones when confronted with higher degrees of demand variations.

[26] use several agents in a common simulated environment, each representing a member of a supply chain. The agents can choose from which market to order. [27] also model several agents in a beer game use-case. Each agent only has limited information on the environment. The authors show that RL policies outperform human-like ones. [28] build on the previous work. They apply DRL to the beer game, showing event greater performance. Apart from using deep neural networks, the main difference is that information is now shared across agents in their model.

Several types of decisions coexist in the literature. The authors cited previously mostly deal with reordering quantities or combinatorial decisions, but other contributions allow choosing between different policies [29]. [30] enables hierarchy in decisions, though their contribution was never directly applied to supply chains.

2.3. Synthesis and research question

Connected supply chain decision-making and RL share strong common properties. They both represent agents immersed in a dynamic and uncertain environment. They also both put the emphasis on keeping track of the environment state at all times to make their decisions. Since they are so closely related, especially through the digital twin paradigm, can we establish a common framework to streamline the application of RL or DRL for connected supply chains?

3. Methods

To combine Reinforcement Learning (RL) and a digital twin on a supply chain problem, we propose a two-step procedure, shown in Figure 1. Each step is described in the next subsections.

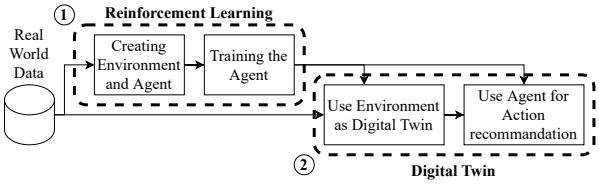

Figure 1. Using the framework in a decision process

The first step is the RL phase, represented by branch 1 in the Figure. The Environment must be created based on the real world problem along with the Agent (or Agents). The Agent is then trained to solve the problem. When training reaches sufficient performance, we switch to branch 2 and the digital twin. The Environment is no longer used to simulate the effects of the Agent's decisions but to reflect the state of the real world problem, as a digital twin.

The second step is the digital twin phase, represented by branch 2 in the Figure. The Agent is used to recommend actions based on the digital twin. It is up to the decision-maker to follow the recommendations or not.

This two-phase approach has the key features identified by [17]. It creates an integrated system which acts as a virtual clone of the real system, ensures data connections to the real system and has simulation and prediction capabilities. The next two subsections describe these two branches.

3.1. Step 1: creating the Environment and training the Agent

Agent training is modeled in Figure 2. It regroups: choosing a model transformer, creating an Environment, a reward method and Agents. They are all described below.

At the start, it is necessary to build the digital twin of the real system, called the Environment. This is done through a **model transformation** scheme by using mathematical modeling, multi-agent simulation or process-mining data collected on the field, for example. For model transformation, no particular technique is recommended in our framework apart from being able to define assets (physical or digital, represented by factory icons) that live in a shared Environment.

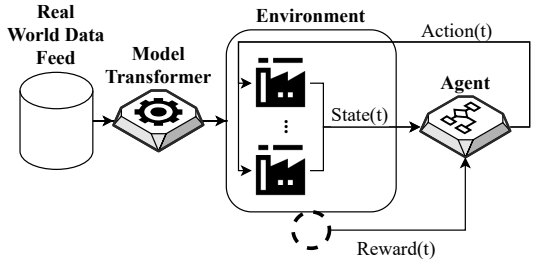

Figure 2. Framework for model training

The **Environment** must have at least three features: (i) collect the $State$ at a time step t, (ii) compute a $Reward$ associated to said $State$ at t and (iii) be able to update its $State$ according to $Actions$. The $State$ method is tasked with collecting all necessary data from

the assets. The meaningful data contained in $State$ will vary from problem to problem. Examples taken from the previous literature range from current inventory position and on-order pipe to vehicle GPS locations.

The $Reward$ method is generally custom-made. It represents the objective function that must be maximized. Rewards are computed based on a $State$ and are made available directly from the Environment as an interface, represented by the dotted circle in Figure 2. Updating the Environment may be done through various means. Depending on Environment complexity, vectorizing the problem in a set of matrices may suffice, whereas more complex cases warrant for the use of discrete event simulation.

The **Agent** is the part that makes decisions to solve the problem. As shown previously in section 2, one may define a single or multiple agents depending on the use-case. The Agent interacts with the Environments over multiple time steps and, possibly multiple episodes (if the problem can be defined by a starting and finishing states). During an interaction step, the Agent first collects the $State(t)$ and outputs an $Action(t)$, according to a $Policy$ it is learning. The Environment registers the $Action(t)$. The effects of this action are taken into account and the Environment advances by one time step. The Environment sends back the $Reward(t+1)$ to the Agent, as well as the new $State(t+1)$ and the loops continue until some ending criterion is met. Any $Action$ available to the Agent is a possible supply chain decision. These information form what is called an *Experience tuple* and have the form $(State(t), Action(t), Reward(t+1), State(t+1))$.

Training the Agent is done by repeating the interaction loop, while improving the performance of the Agent's $Policy$. Multiple learning algorithms can be used but in our DRL approach, we focus on using deep neural networks to serve as policies. Choosing the correct algorithm for learning also derives from several criteria: (i) complexity of implementation, (ii) discrete or continuous States and (iii) discrete or continuous Actions. Training stops with different reasons: (i) sufficient performance has been reached, (ii) the maximum allowed training time has been consumed or (iii) early stopping is enforced when performance has stopped increasing for a set number of episodes.

3.2. Step 2: using the Environment as the digital twin

Using the Environment and Agent as a digital twin starts with establishing a model adapter, as described in Figure 3. The adapter may be of any kind (communication layer of a digital twin, digital copy of a physical internet network, for example). It is meant to transform the raw data from real world feeds into a $State$ usable by the Agent.

When using the Environment and Agent as a digital twin, training is deactivated. We only wish to exploit the Agent's policy in this case. As a result, there are no training loops or episodes needed. Instead, the Agent directly recommends Actions based on its learned policy. The $Actions$ are then sent to an adapter which, in turn, transforms them into real world instructions. The two adapters may be the same and allow for two-way communication. In this step, the Environment actually *becomes* an equivalent to the digital twin and the Agent *replaces* the human in charge of evaluating the different possibilities.

We must note that it is still possible to learn at this stage, even though it may be detrimental to the Agent's performance. When used as a digital twin, the Environment still keeps track of Experience tuples. As such, learning can be reactivated either fully or at selected points in time (every week for example). As training deep neural networks may result in lower performance, one must exercise caution before pushing the newly trained network into production when using it as a digital twin.

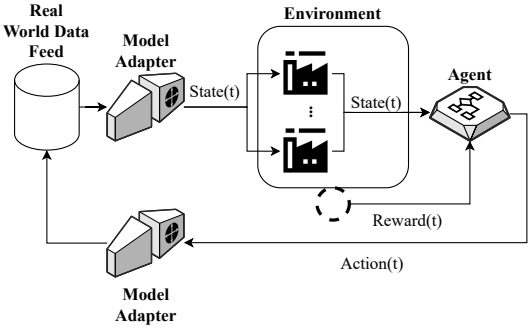

Figure 3. The Environment serves as the digital twin

4. Use-case and data

In order to test our framework, we apply it to a drug supply chain instance. Our version is based on the works of [28] on the Beer Game, the works of [31] for translating inventory management problems into DRL models and governmental rules for drug-specific aspects in [32]. Preventing shortages on the drug supply chain is a major health and governmental issue, even more so under pandemic conditions [33].

4.1. Use-case specifications

Our supply chain is divided into four actor classes, named at the governmental level as [34]: (i) *Pharmaceutical establishments* who manufacture, import and sell drugs, (ii) *Custodians* who distribute drugs, (iii) *Wholesaler-distributors* who resell drugs to pharmacies and (iv) *Pharmacies* or *Internal pharmacies* who act as retailers, respectively outside and inside hospitals.

Of these four classes, three of them have to place orders upstream at the end of each period, with a common objective of minimizing the total supply chain costs. The pharmaceutical establishments have access to unlimited raw materials and may place an order of any size. These four members form the Environment.

Each member is separated by fixed lead times. These are different between each stage but do not change over time. The maximum lead time is called max_{LT}. Each member i also has a given production capacity C_i. Any order exceeding capacity is automatically capped and the remainder is forgotten. The retailer receives stochastic demands in the form of a normal distribution of known parameters (μ, σ). In this, we deviate from classic works, for example on the Beer Game, but this is done to make the use-case slightly more realistic. It is also done to make it harder for the Agent to learn the real nature of the demand. Note that members may also have an order backlog that will need to be fulfilled before new orders.

The total reward function reflects the total supply chain costs. Each member has: (i) a sell price p_i, (ii) holding costs b_i and (iii) stock-out costs h_i. These parameters are taken from the real world data. The total Reward over the i members of the supply chain at time t is

$$Reward = \sum_i S_i * p_i - (RO_i * r_i + B_i * b_i + IL_i * h_i)$$

where S_i is for sold units at stage i, RO_i for replenishment orders sizes, IL_i for inventory levels and b_i for total backlog.

In our use case, model transformation is done through mathematical modeling of the problem. We follow [31] and use a set of matrices to define the states of all members of the supply chain and the state of the supply lines between the members.

Contrary to [28], we solve the problem using a single Agent aware of the full state of the supply chain. This represents the centralized view that is obtained in a digital twin paradigm. The next step is to define the State, the Actions and how the Agent learns its policy.

Following [31], the State contains the inventory levels (and not the inventory positions) of the first three members and the replenishment order quantities passed over the last max_{LT} periods. As a result, State is a vector of size $max_{LT} + 3$. The manufacturer is not included as it has infinite supply capacity and no lead time.

The first role of the Agent is to build a policy over time. This policy should, at minimum, take in the State and output the Action required by the Environment. When using DRL, policies are represented by using different neural network architectures. Our use-case makes no exception: the Agent neural network follows the *Action Branching Architecture* from [35]. An overview of the architecture is given in Figure 4. The choice of this architecture is motivated by both the structure of the supply chain and having a single Agent. Having a single Agent means that it is responsible for deciding the individual Action (the replenishment order size) of each member i. The Agent then issues three different actions at each period. The action space of a member i is discrete and defined between 0 and C_i. The Action Branching Architecture allows for this.

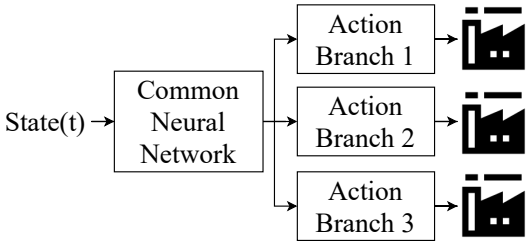

Figure 4. Action Branching Architecture overview

In the Action Branching Architecture, the State first passes through a common neural network that is used to share the same representation of the information between all members. Each member i then has its own dedicated action branch. Each branch outputs an Action and all Actions and transmitted to the Environment. All possible Actions for member i belong to an action space called $\mathcal{A}_i$. In the case where members have different capacities, a technique called *masking* is used. This allows to create action branches that all have the same output dimensions (the maximum capacity C_{max}) thus making for easier training. Masking disables the illegal outputs of each branch without modifying the network's structure.

Each branch outputs an individual Q-value for an action a in a state s, denoted $Q_i(s, a)$. The common network and action branches contain streams of data that compute the value of a state $V(s)$ and the individual action advantages $A_i(s, a)$. These streams are not

represented in Figure 4 for brevity but the reader may refer to [35] for further details. Based on the works of [35] and [36], we defined the value for member i to take action a in state s as

$$Q_i(s,a) = V(s) + (A_i(s,a) - \frac{1}{n}\sum_{a' \in \mathcal{A}_i} A_i(s,a'))$$

The common network follows the works of [35]. It uses an architecture called Dueling Double Deep Q-Network (Dueling DDQN), taken from [36]. A Deep Q-Network (DQN) progressively learns the value of taking an action in a given state [37]. In the RL field, this value is called the Q-value [10], hence the name DQN. The *Double* term refers to the use of twin networks inside the Agent. One of them is called the *online network*, the second is called the *target network* [38]. The role of these networks is explained in subsection 4.2. The *Dueling* aspect is the addition of a new stream in the network to evaluate the *Advantage* and not only the Q-value. The Advantage can be understood as computing the interest of not following the current best Q-value in the policy. Practically, it makes for faster and more stable training.

We also include two other agents that do not implement any RL algorithm, in order to compare numerical results. These agents act as baselines for two cases: (i) a base-stock agent represents the use of DT with shared information but strong hypotheses on demand distribution and no learning capability and (ii) a human-like agent represents the absence of shared information and learning. The first agent uses a base-stock policy with safety stocks fitted to the same distribution $\mathcal{N}(\mu, \sigma)$ we used for the neural network agent. We aim for a 95% service level to set safety stock size. This agent serves as a baseline and has been shown to obtain good performance on the problem [39]. The second agent is called *Human-like* and is based on the heuristic described in [40]. The Human-like agent is based on practitioner feedback and tries to maintain a desired stock level and a desired supply level throughout the periods.

4.2. Agent training

Agent training is then a matter of optimizing the neural network on experience tuples. To do so, we run several episodes of simulation on the Environment. An episode in our use-case is composed of 30 periods of the Beer Game. Each period is an interaction step between the Agent and the Environment during which the Agent gather an experience tuple. Tuples are kept in the Agent's memory.

The Agent's policy is improved by selecting batches of experiences and feeding them to the neural networks. Feeding experiences to the Agent's target network provides a target to optimize against. Feeding experiences to the Agent's online network gives the current performance of the Agent. Optimization is then a matter of updating the online network's weights according to the error between the two outputs. After an optimization step, the weights of the online network are copied onto the target network.

The error between the two outputs is called the temporal difference target [10]. For each member i, we define the individual error as

$$y_i = Reward + \gamma Q_i^-(s, \underset{a' \in \mathcal{A}_i}{\operatorname{argmax}} Q_i(s,a'))$$

The overall error is defined as

$$y = Reward + \gamma \frac{1}{N}\sum_i Q_i^-(s, \underset{a' \in \mathcal{A}_i}{\operatorname{argmax}} Q_i(s,a'))$$

where N is the total number of agents that need to place orders upstream (3 in our use-case), Q_i^- is the target network function and γ a discount factor (rewards have more weight if they are obtained early in the episodes).

Keeping two networks helps with training the Agent. We must note that training the Agent's neural network is quite different from training a regular neural network. When using DRL, we do not train directly on real world data, as would be the case for a regular network. Instead, we feed data from the Environment. As such, we do not have any ground truth to optimize against and use the target network as a stable 'synthetic' target.

At the end of each episode, the Environment is reset. To obtain a more general policy, we use 5 different random seeds. As a result, when changing seeds, the environment is reset to a different state. We used three criteria to stop the training: (i) having played 10000 episodes total, (ii) having reached a average reward of 2000 points over the last 100 episodes or (iii) having trained for 30 minutes.

4.3. Using the Agent's policy after training

Once training is complete, we can start exploiting the Agent's policy to produce action recommendations. We start with model adaptation. It is relatively simple in this use-case: we hook up a data generator to the Environment. The generator represents a data feed similar to what was seen in training, only with different

Usage	Mean			Max			Min			Std		
	BS	HL	BA	BS	HL	BA	BS	HL	BA	BS	HL	BA
Training	996	-571	1006	1002	-537	1452	990	-615	57	1.39	8.67	280
Evaluation	996	-572	585	1009	-500	1420	992	-596	-636	1.35	7.97	428

Table 1. Training and using the agent result summary

random seeds. The generator has two streams: (i) states show the state of each member of the supply chain (the rules for updating the state are the ones from the Beer Game), and (ii) demands for the retailer that follow $\mathcal{N}(\mu, \sigma)$ we used for training.

To evaluate the Agent's performance, we simulate 100 episodes of 30 periods, where we will always follow its recommended actions. During evaluation, the Agent can no longer learn.

Performance is measured through the moving average of the Agent's reward. We used a window of 100 episodes. We added the envelope of the performance represented by the minimum and maximum reward values of the episode. It is useful to assess the deviations of the recommended actions. We compare the results of our Agent, the Base-stock agent and the Human-like agent in the next section.

5. Results

All the results are gathered in Figure 5. The upper half shows the evolution of the performance during the Agent's training. The lower half represents the performance during evaluation, when we used the Agent to recommend actions. Time is on the x axis with the growing number of episodes. Performance is on the y axis. A summary is given in Table 1 with variants identified as "BS" for Basestock, "HL" for Human-like and "BA" for Branching actions.

Our Agent's performance is figured by the red dotted lines and envelopes and is called Branching Actions. By comparison, the Base-stock agent is in black and the human-like agent in blue. We can see first that the human-like agent where each member knows only about its own state has the lowest performance. The Base-stock agent, as said previously in Section 3 reaches good performance of around 1000 points. Our Agent slowly comes to the Base-stock level of performance and sometimes outperforms it on average or in maximum.

By comparing the red areas to the black and blue lines, we see however that our Agent is still the less stable of the three. This is mostly due to the neural network training procedure. It can be dampened by switching to other network architectures which may be more complex. It can also be reduced by finding better hyper-parameters such as adaptable learning rates or batch sizes.

As a whole these results first show that it is feasible to use a RL Agent and Environment scheme in conjunction with a DT of a supply chain. The much lower performance of the human-like reinforce the need for communication between members. The fact that our RL Agent comes close to the base-stock agent and even outperforms it in the end shows that adding learning capabilities bears fruit. We must also note that the base-stock agent was used in its perfect setting, using exactly the same distributions. In reality, data would seldom follow such well-defined patterns however, and our RL Agent may have been able to achieve event better results.

6. Conclusion

Through this paper, we developed two main contributions. First, we showed how the fields of connected supply chain management and reinforcement learning could meet by sharing a common framework. Second, we shared some promising results on applying DRL inside this framework to recommend actions.

The framework we proposed lends itself to several adaptations. It was derived with a DT focus in the use-case but it is not limited to this. A physical internet based system could be translated in terms of Environment (the complete list of all physical and digital assets) and Agents. Furthermore, if the Agents of our case live in the margins of the Environment, they can be more involved. Fleets of vehicles can be considered as Agents with or without another central Agent to guide them.

With this framework and a simple use-case, we demonstrated how to give advanced learning capabilities to digital twins. The common features between the DT and RL paradigms provide a natural way to control supply chain systems. Furthermore, the first results are encouraging and hint at the possibility of expanding applications: larger problems, complex multiple agents systems or decentralized decision-making, for instance. Another implication resides in the possibility to easily swap components and frameworks: efficient RL agents could be used in Cyber-Physical Systems, provided they operate with the same "State".

The Dueling DDQN we used to learn the Agent's policy is also just a first step towards an efficient and

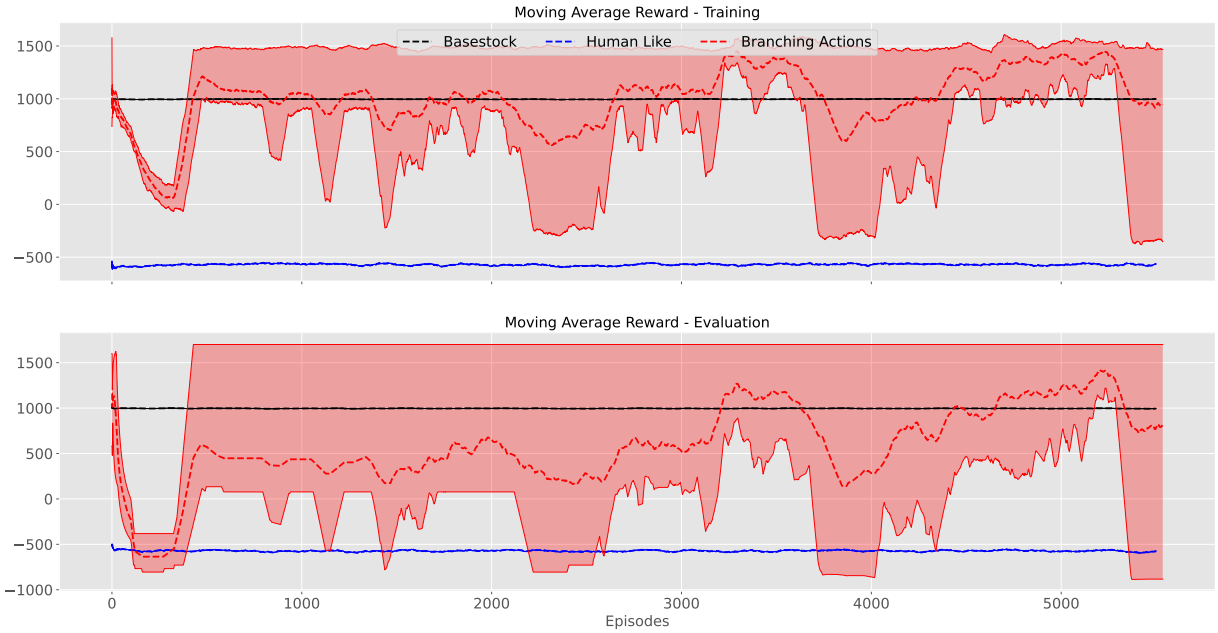

Figure 5. Training and using the agent over time

connected control of a supply chain. State-of-the-art in DRL goes further than this network architecture. Recent contributions allow complex types of inputs such as graphs or images and also make better use of experience tuples, thus allowing for faster training.

This paper is merely a first step towards a streamlined control of supply chains systems. Next steps involve, for example, the generation of complex Environments from knowledge databases or pre-calibrated digital twins in the model transformer step. At the Environment level, being able to craft specific rewards and states for different types of problems is also an interesting avenue of research. It could lead to faster convergence during training of the more complex models. Our results should also be confronted to other frameworks (Cyber-Physical System with routing algorithms, DTs with human control) on the same use-case.

References

[1] M. Christopher, *Logistics & supply chain management*. Pearson Uk, 2016.

[2] A.-M. Barthe-Delanoë, S. Truptil, F. Bénaben, and H. Pingaud, "Event-driven agility of interoperability during the run-time of collaborative processes," *Decision Support Systems*, vol. 59, pp. 171–179, 2014.

[3] A. Fertier, G. Martin, A.-M. Barthe-Delanoë, J. Lesbegueries, A. Montarnal, S. Truptil, F. Bénaben, and N. Salatgé, "Managing events to improve situation awareness and resilience in a supply chain," *Computers in industry*, in press.

[4] D. Ivanov, B. Sokolov, and A. Dolgui, "The ripple effect in supply chains: trade-off 'efficiency-flexibility-resilience'in disruption management," *International Journal of Production Research*, vol. 52, no. 7, pp. 2154–2172, 2014.

[5] A. Sarac, N. Absi, and S. Dauzère-Pérès, "A literature review on the impact of rfid technologies on supply chain management," *International journal of production economics*, vol. 128, no. 1, pp. 77–95, 2010.

[6] D. A. Greenwood, C. Dannegger, K. Dorer, and M. Calisti, "Dynamic dispatching and transport optimization-real-world experience with perspectives on pervasive technology integration.," in *hicss*, pp. 1–9, 2009.

[7] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[8] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design automation conference*, pp. 731–736, IEEE, 2010.

[9] D. Ivanov and A. Dolgui, "A digital supply chain twin for managing the disruption risks and resilience in the era of industry 4.0," *Production Planning & Control*, pp. 1–14, 2020.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] B. Montreuil, "Toward a physical internet: meeting the global logistics sustainability grand challenge," *Logistics Research*, vol. 3, no. 2, pp. 71–87, 2011.

[12] H. Treiblmaier, K. Mirkovski, P. B. Lowry, and Z. G. Zacharia, "The physical internet as a new supply chain paradigm: a systematic literature review and a comprehensive framework," *The International Journal of Logistics Management*, 2020.

[13] E. Ballot, B. Montreuil, and R. Meller, *The physical internet*. La Documentation Française, 2014.

[14] Y. Sallez, S. Pan, B. Montreuil, T. Berger, and E. Ballot, "On the activeness of intelligent physical internet containers," *Computers in Industry*, vol. 81, pp. 96–104, 2016.

[15] X. T. Kong, X. Yang, K. Peng, and C. Z. Li, "Cyber physical system-enabled synchronization mechanism for pick-and-sort ecommerce order fulfilment," *Computers in Industry*, vol. 118, p. 103220, 2020.

[16] Y. Yang, S. Pan, and E. Ballot, "Innovative vendor-managed inventory strategy exploiting interconnected logistics services in the physical internet," *International Journal of Production Research*, vol. 55, no. 9, pp. 2685–2702, 2017.

[17] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167653–167671, 2019.

[18] S. Singh, A. Barde, B. Mahanty, and M. Tiwari, "Digital twin driven inclusive manufacturing using emerging technologies," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2225–2230, 2019.

[19] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in cps-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017.

[20] C. Fan, C. Zhang, A. Yahja, and A. Mostafavi, "Disaster city digital twin: A vision for integrating artificial and human intelligence for disaster management," *International Journal of Information Management*, vol. 56, p. 102049, 2021.

[21] J. Cirullies and C. Schwede, "On-demand shared digital twins–an information architectural model to create transparency in collaborative supply networks," in *Proceedings of the 54th Hawaii International Conference on System Sciences*, p. 1675, 2021.

[22] H. Liang, X. Wen, Y. Liu, H. Zhang, L. Zhang, and L. Wang, "Logistics-involved qos-aware service composition in cloud manufacturing with deep reinforcement learning," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 101991, 2021.

[23] N. Vanvuchelen, J. Gijsbrechts, and R. Boute, "Use of proximal policy optimization for the joint replenishment problem," *Computers in Industry*, vol. 119, p. 103239, 2020.

[24] E. Puskás, Á. Budai, and G. Bohács, "Optimization of a physical internet based supply chain using reinforcement learning," *European Transport Research Review*, vol. 12, no. 1, pp. 1–15, 2020.

[25] H. D. Perez, C. D. Hubbs, C. Li, and I. E. Grossmann, "Algorithmic approaches to inventory management optimization," *Processes*, vol. 9, no. 1, p. 102, 2021.

[26] A. Aghaie and M. Hajian Heidary, "Simulation-based optimization of a stochastic supply chain considering supplier disruption: Agent-based modeling and reinforcement learning," *Scientia Iranica*, vol. 26, no. 6, pp. 3780–3795, 2019.

[27] T. van Tongeren, U. Kaymak, D. Naso, and E. van Asperen, "Q-learning in a competitive supply chain," in *2007 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1211–1216, IEEE, 2007.

[28] A. Oroojlooyjadid, M. Nazari, L. Snyder, and M. Takác, "A deep q-network for the beer game: A reinforcement learning algorithm to solve inventory optimization problems," *arXiv preprint arXiv:1708.05924*, 2017.

[29] D. Chodura, P. Dominik, and J. Koźlak, "Market strategy choices made by company using reinforcement learning," *Trends in Practical Applications of Agents and Multiagent Systems*, pp. 83–90, 2011.

[30] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*, pp. 3540–3549, PMLR, 2017.

[31] C. D. Hubbs, H. D. Perez, O. Sarwar, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, "Or-gym: A reinforcement learning library for operations research problem," *arXiv preprint arXiv:2008.06319*, 2020.

[32] L'Assemblée nationale and Le Président de la République, "Loi relative au renforcement de la sécurité sanitaire du médicament et des produits de santé," Dec. 2011.

[33] MSS, "Rupture d'approvisionnement d'un médicament," Sept. 2016.

[34] MSS, "Le circuit de distribution du médicament en France," Jan. 2017.

[35] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[36] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, pp. 1995–2003, PMLR, 2016.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[38] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[39] A. J. Clark and H. Scarf, "Optimal policies for a multi-echelon inventory problem," *Management science*, vol. 6, no. 4, pp. 475–490, 1960.

[40] J. D. Sterman, "Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment," *Management science*, vol. 35, no. 3, pp. 321–339, 1989.