






Multi-Agent Reinforcement Learning Tool for Job Shop Scheduling Problems

Yailen Martínez Jiménez¹ ^(✉), Jessica Coto Palacio² , and Ann Nowé³ 

¹ Universidad Central “Marta Abreu” de Las Villas, Santa Clara, Cuba
yailenm@uclv.edu.cu

² UEB Hotel Los Caneyes, Santa Clara, Cuba
informatico@caneyes.vcl.tur.cu

³ Vrije Universiteit Brussel, Brussels, Belgium
ann.nowe@ai.vub.ac.be

Abstract. The emergence of Industry 4.0 allows for new approaches to solve industrial problems such as the Job Shop Scheduling Problem. It has been demonstrated that Multi-Agent Reinforcement Learning approaches are highly promising to handle complex scheduling scenarios. In this work we propose a user friendly Multi-Agent Reinforcement Learning tool, more appealing for industry. It allows the users to interact with the learning algorithms in such a way that all the constraints in the production floor are carefully included and the objectives can be adapted to real world scenarios. The user can either keep the best schedule obtained by a Q-Learning algorithm or adjust it by fixing some operations in order to meet certain constraints, then the tool will optimize the modified solution respecting the user preferences using two possible alternatives. These alternatives are validated using OR-Library benchmarks, the experiments show that the modified Q-Learning algorithm is able to obtain the best results.

Keywords: Reinforcement Learning · Multi-Agent Systems · Industry 4.0 · Job Shop Scheduling

1 Introduction

During the last years, technological developments have increasingly benefited industry performance. The appearance of new information technologies have given rise to intelligent factories in what is termed as Industry 4.0 (i4.0) [11, 12]. The i4.0 revolution involves the combination of intelligent and adaptive systems using shared knowledge among diverse heterogeneous platforms for computational decision-making [11, 13, 21], within Cyber-Physical Systems (CPS). In this sense, embedding Multi-Agent Systems (MAS) into CPS is a highly promising approach to handle complex and dynamic problems [13]. A typical example of an industrial opportunity of this kind is scheduling, whose goal is to achieve resource optimization and minimization of the total execution time [19]. Given the complexity and dynamism of industrial environments, the resolution of this type of

problem may involve the use of very complex solutions, as customer orders have to be executed, and each order is composed by a number of operations that have to be processed on the resources or machines available. In real world scheduling problems, the environment is so dynamic that all this information is usually not known beforehand. For example, manufacturing scheduling is subject to constant uncertainty, machines breakdown, orders take longer than expected, and these unexpected events can make the original schedule fail [10, 24].

Accordingly, the problem of creating a job-shop scheduling, known as Job-Shop Scheduling Problem (JSSP), is considered one of the hardest manufacturing problems in the literature [1]. Many scheduling problems suggest a natural formulation as distributed decision making tasks. Hence, the employment of MAS represents an evident approach [5]. These agents typically use Reinforcement Learning (RL), which is learning what to do (how to map situations to actions) so as to maximize a numerical reward signal [18]. It allows an agent to learn optimal behavior through trial-and-error interactions with its environment. By repeatedly trying actions in different situations the agent can discover the consequences of its behavior and identify the best action for each situation. For example, when dealing with unexpected events, learning methods can play an important role, as they could ‘learn’ from previous results and change specific parameters for the next iterations, allowing not only to find good solutions, but more robust ones.

Another problem that has been identified in the scheduling community is the fact that most of the research concentrates on optimization problems that are a simplified version of reality. As the author points out in [20]: “this allows for the use of sophisticated approaches and guarantees in many cases that optimal solutions are obtained. However, the exclusion of real-world restrictions harms the applicability of those methods. What the industry needs are systems for optimized production scheduling that adjust exactly to the conditions in the production plant and that generate good solutions in very little time”. In this research we propose a Multi-Agent Reinforcement Learning tool that allows the user to either keep the best result obtained by a learning algorithm or to include extra constraints of the production floor. This first version allows to fix operations to time intervals in the corresponding resources and afterwards optimize the solution based on the new constraints added by the user. This is a first approach that helps to close the gap between literature and practice.

2 Literature Review

As it has been mentioned before, scheduling is a decision-making process concerned with the allocation of limited resources (machines, material handling equipment, operators, tools, etc.) to competing tasks (operations of jobs) over time with the goal of optimizing one or more objectives [15]. The output of this process is time/machine/operation assignments [9]. Scheduling is considered as one of the key problems in manufacturing systems, and it has been a subject of interest for a long time. However, it is difficult to talk about a method that gives optimal solutions for every problem that emerges [2].

Different Operations Research (OR) techniques (Linear Programming, Mixed-Integer Programming, etc.) have been applied to scheduling problems. These approaches usually involve the definition of a model, which contains an objective function, a set of variables and a set of constraints. OR based techniques have demonstrated the ability to obtain optimal solutions for well-defined problems, but OR solutions are restricted to static models. Artificial Intelligence approaches, on the other hand, provide more flexible representations of real-world problems, allowing human expertise to be present in the loop [8].

2.1 Job Shop Scheduling

A well-known manufacturing scheduling problem is the classical JSSP, which involves a set of jobs and a set of machines with the purpose of finding the best schedule, that is, an allocation of the operations to time intervals on the machines that has the minimum duration required to complete all jobs (in this case the objective is to minimize the makespan). The total number of possible solutions for a problem with n jobs and m machines is $m(n!)$. In this case, exact optimization methods fail to provide timely solutions. Therefore, we must turn our attention to find methods that can efficiently produce satisfactory (but not necessarily optimal) solutions [14]. Some of the restrictions inherent in the definition of the JSSP are the following:

- Only one operation from each job can be processed simultaneously.
- No preemption (i.e. process interruption) of operations is allowed.
- Each job must be processed to completion and no job is processed twice on the same machine.
- Jobs may be started and finished at any time, i.e., no release or due dates times exist.
- Machines cannot process more than one operation at a time.
- There is only one machine of each type and they may be idle within the schedule period.
- Jobs must wait for the next machine in the processing order to become available.
- The machine processing order of each job is known in advance and it is immutable.

Operations Research offers different mathematical approaches in order to solve scheduling problems, for example Linear Programming, Dynamic Programming and Branch and Bound methods. When the size of the problem is not too big, these methods can provide optimal solutions in a reasonable amount of time. Most real world scheduling problems are NP-hard, and the size is usually not small, that is why optimization methods fail to provide optimal solutions in a reasonable timespan. This is where heuristic methods become the focus of attention, these methods can obtain good solutions in an efficient way. Artificial Intelligence became an important tool to solve real world scheduling problems in the early 80s [26]. In [5, 6], the authors suggested and analyzed the application of

RL techniques to solve job shop scheduling problems. They demonstrated that interpreting and solving this kind of scenarios as a multi-agent learning problem is beneficial for obtaining near-optimal solutions and can very well compete with alternative solution approaches.

2.2 Multi-Agent Reinforcement Learning (MARL)

The Reinforcement Learning paradigm is a popular way to address problems that have only limited environmental feedback, rather than correctly labeled examples, as is common in other machine learning contexts. While significant progress has been made to improve learning in a single task, the idea of transfer learning has only recently been applied to reinforcement learning tasks. The core idea of transfer is that experience gained in learning to perform one task can help improve learning performance in a related, but different, task. There are many possible approaches to learn such a policy, Temporal Difference methods, such as Q-Learning (QL) [18, 22] and Sarsa [7, 16], policy search methods, such as policy iteration (dynamic programming), policy gradient [3, 23], and direct policy search [25], among others. The general idea behind them is to learn through interaction with an environment and the steps can be summarized as follows:

1. The agent perceives an input state.
2. The agent determines an action using a decision-making function (policy).
3. The chosen action is performed.
4. The agent obtains a scalar reward from its environment (reinforcement).
5. Information about the reward that has been received for having taken the recent action in the current state is processed.

The basic RL paradigm is to learn the mapping from states to actions only on the basis of the rewards the agent gets from its environment. By repeatedly performing actions and observing resulting rewards, the agent tries to improve and fine-tune its policy. RL is considered as a strong method for learning in MAS environments. Multi-Agent Systems are a rapidly growing research area that unifies ideas from several disciplines, including artificial intelligence, computer science, cognitive science, sociology, and management science. Recently, there has been a considerable amount of interest in the field motivated by the fact that many real-world problems such as engineering design, intelligent search, medical diagnosis, and robotics can be best modeled using a group of problem solvers instead of one, each named agent [17].

3 Multi-Agent Reinforcement Learning Tool

The MARL tool groups several algorithms aimed at solving scheduling problems in the manufacturing industry. This paper proposes a first version of a tool which focuses on the need of building a more flexible schedule, in order to adjust it to the user's requests without violating the restrictions of the JSSP scenario.

Figure 1 shows the main interface, where the user must first choose the file where the information related to the problem is described, basically the jobs that need to be processed, the resources available to execute them and the processing times (open button). The original algorithms are based on solving the JSSP.

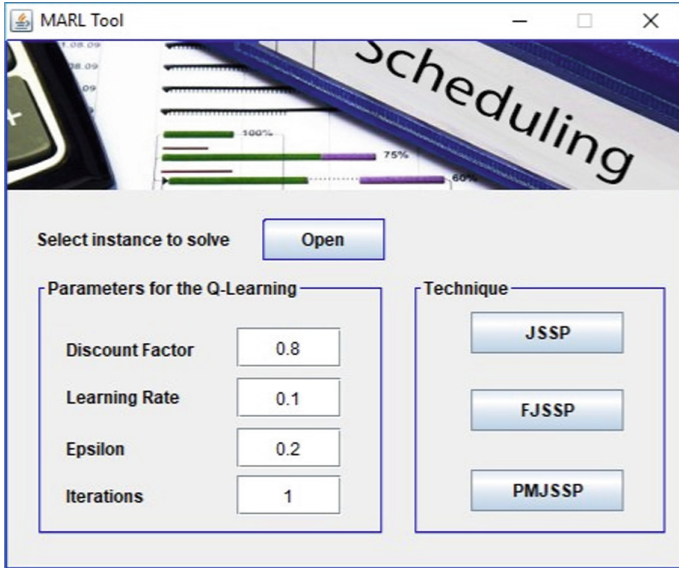


Fig. 1. Main interface of the MARL tool.

The approach used to obtain the original solution that the user can afterwards modify is the one proposed in [14], it is a generic multi-agent reinforcement learning approach that can easily be adapted to different scheduling settings, such as the Flexible Job Shop (FJSSP) or the Parallel Machines Job Shop Scheduling (PMJSSP). The algorithm used is the Q-Learning, which works by learning an action-value function that gives the expected utility of taking a given action in a given state. There is basically an agent per machine which takes care of allocating the operations that must be executed by the corresponding resource. Figure 2 shows the agents in a scheduling environment, and the parameters on the left of the main interface are explained in detail in [14].

Once the user chooses the scheduling scenario to solve (JSSP, FJSSP or PMJSSP), the tool proposes an initial solution (Fig. 3) based on the original QL algorithm described before, and at the same time it enables a set of options that are the basis of this research. The user has the possibility to move the operations either using the mouse or the touch screen, and these movements must be validated once the new positions are decided.

All the options are explained in detail below:

- **Save Schedule:** It allows to save the schedule as an image (.png) through a dialog box to choose the path and to specify the file name.

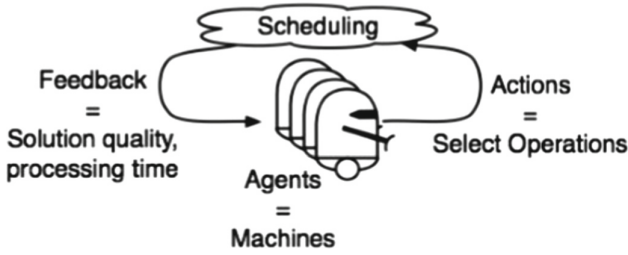


Fig. 2. Agents in a scheduling environment, as proposed in [16].

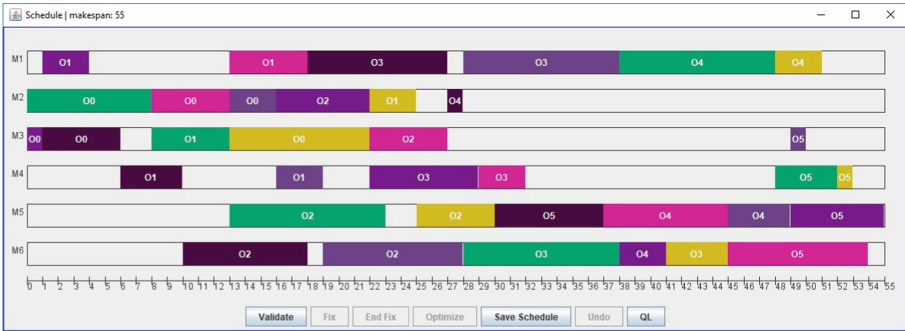


Fig. 3. Example of a schedule obtained using the MARL tool for the ft06 instance.

- **Validate:** Once an operation is moved from its original position, the new schedule must be validated either with a right or a left shifting so that the tool can then allow to make new changes. If the start time of an operation is increased (it is shifted to the right), then the start time of the next operation of the same job is checked and if it starts before the new end time of the previous one, adjustments to the schedule have to be made. As a consequence, the first thing is to aspire to locate that operation right after its predecessor, in case the new placement obstructs the processing of another operation in the same resource, the new start time becomes the end time of that other operation, and so on, the possible locations are analyzed until an available time slot is found. The shift to the left occurs similarly with the exception that the operation is placed in such a way that its execution starts earlier. The algorithm always checks that it is a valid movement, that is, that it does not start before the minimum possible start time for that operation. Regarding the following operations of the same job, the algorithm tries to move them as close as possible to their predecessor, in order to minimize the makespan.
- **Fix:** This option is enabled once the new schedule is validated, in order to optimize afterwards the schedule with the new changes. The fixed operations are highlighted in black and there is the possibility of pressing them again to stop fixing their position.

- **End Fix:** The user has to choose this option once the process of fixing the operations is finished, and then proceed to optimize the schedule, either using the shiftings or using the Q-Learning algorithm.
- **Optimize:** After fixing the operations that the user wants to keep in the specified positions, then the rest of the schedule can be optimized. This is based on performing a left shift on all the movable operations, respecting the constraints of the job shop scheduling and also the start times of the fixed operations. The procedure is performed according to the position of the operations on the x-axis, in increasing order according to their starting times. When an operation different from the first of each job is selected, its new initial time will be the end time of its predecessor, if this is not a valid movement because it interferes with the execution of another operation being processed on the same machine, then it is shifted to the first available interval where it fits on that resource, and if this is not possible then it keeps its original position.
- **Q-Learning:** This optimization is based on applying the QL algorithm described before, including a new constraint, in this case the algorithm will learn a schedule taking into account the operations that were fixed by the user.
- **Undo:** It is possible to go back as many schedules as validations have been made.

In this paper we compare the performance of the two alternatives for optimizing the schedule once the user has fixed some operations, the classical left shifting which is executed when clicking the optimize button and the modified Q-Learning version, which includes the position of the fixed operations in the learning process.

4 Experimental Results

In order to measure the performance of the two alternatives several benchmark problems from the OR-Library [4] were used. The OR-Library is a library of problem instances covering various OR problems. Table 1 shows the results for 11 JSSP instances, with different number of jobs and machines.

The column optimum represents the best-known solution for the corresponding instance; Original QL refers to the best solution obtained by the original version of the QL algorithm, without any extra constraints. For the results shown in the last two columns some modifications were made to the solution obtained by the original QL, for each instance the same operations were fixed, and each optimization alternative had to adjust the schedule in order to minimize the makespan. To determine if there are significant differences in the results obtained by the alternatives a statistical analysis was performed and the results are shown in Fig. 4.

As it can be seen, the Wilcoxon test shows that there are significant differences between the two alternatives ($\text{sig} = 0.08$), the mean ranks confirm that the

Table 1. Experimental results using instances from the OR-Library.

Instance	Optimum	Original QL	Optimize	QL with fixed operations
ft06	55	55	82	76
la01	666	666	849	810
la02	655	667	848	801
la03	597	610	752	730
la04	590	611	647	640
la05	593	593	603	603
la06	926	926	1012	1008
la07	890	890	1016	1010
la08	863	863	1060	1043
la09	951	951	1144	1096
la10	958	958	1016	1016

		N	Mean Rank	Sum of Ranks
QL_modif - Optimize	Negative Ranks	9 ^a	5,00	45,00
	Positive Ranks	0 ^b	,00	,00
	Ties	2 ^c		
	Total	11		

a. QL_modif < Optimize, b. QL_modif > Optimize, c. QL_modif = Optimize

	QL_modified - Optimize
Z	-2,668 ^b
Asymp. Sig. (2-tailed)	,008

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

Fig. 4. Statistical analysis using the Wilcoxon test.

QL version with fixed operations is able to obtain better results than the classical optimization process of shifting the operations (optimize). This is mainly because the left shifting respects the order in which the operations were initially placed along the x axis. The QL algorithm, on the other hand, keeps the fixed positions and during the process of learning, the order in which the operations are scheduled in the resources does not have to be the same, this allows the approach to obtain better solutions in terms of makespan.

5 Conclusions

This paper proposed a Multi-Agent Reinforcement Learning tool for the Job Shop Scheduling Problem, which can be adapted to other scheduling scenarios as the Flexible JSSP and the Parallel Machines JSSP. This tool allows the user to keep the best schedule obtained by the original QL algorithm or to make adjustments in order to move operations to fix intervals, according to the constraints of the production floor. After all the adjustments have been made, a rescheduling process is started in order to optimize as much as possible the modified solution. This optimization can be done by shifting to the left all the possible movable operations or using a modified version of the QL algorithm. The alternatives were evaluated using benchmark data from the OR-Library and the results showed that the QL algorithm is able to show the best results.

References

1. Asadzadeh, L.: A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Comput. Ind. Eng.* **85**, 376–383 (2015)
2. Aydin, M.E., Oztemel, E.: Dynamic job-shop scheduling using reinforcement learning agents. *Robot. Auton. Syst.* **33**, 169–178 (2000)
3. Baxter, J., Bartlett, P.L.: Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res.* **15**, 319–350 (2001)
4. Beasley, J.E.: OR-Library: distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41**(11), 1069–1072 (1990)
5. Gabel, T.: Multi-agent reinforcement learning approaches for distributed job-shop scheduling problems. Ph.D. thesis, Universität Osnabrück (2009)
6. Gabel, T., Riedmiller, M.: On a successful application of multi-agent reinforcement learning to operations research benchmarks. In: *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, Honolulu, pp. 68–75 (2007)
7. Gavin, R., Niranjana, M.: On-line Q-learning using connectionist systems. Technical report, Engineering Department, Cambridge University (1994)
8. Gomes, C.P.: Artificial intelligence and operations research: challenges and opportunities in planning and scheduling. *Knowl. Eng. Rev.* **15**(1), 1–10 (2000)
9. Goren, S., Sabuncuoglu, I.: Robustness and stability measures for scheduling: single-machine environment. *IIE Trans.* **40**(1), 66–83 (2008)
10. Hall, N., Potts, C.: Rescheduling for new orders. *Oper. Res.* **52**, 440–453 (2004)
11. Leitao, P., Colombo, A., Karnouskos, S.: Industrial automation based on cyber-physical systems technologies: prototype implementations and challenges. *Comput. Ind.* **81**, 11–25 (2016)
12. Leitao, P., Rodrigues, N., Barbosa, J., Turrin, C., Pagani, A.: Intelligent products: the grace experience. *Control Eng. Pract.* **42**, 95–105 (2005)
13. Leusin, M.E., Frazzon, E.M., Uriona Maldonado, M., Kück, M., Freitag, M.: Solving the job-shop scheduling problem in the industry 4.0 era. *Technologies* **6**(4), 107 (2018)
14. Martínez Jiménez, Y.: A generic multi-agent reinforcement learning approach for scheduling problems. Ph.D. thesis, Vrije Universiteit Brussel, Brussels (2012)

15. Pinedo, M.: Scheduling: Theory, Algorithms and Systems. PrenticeHall, Englewood cliffs (1995)
16. Singh, S., Sutton, R.S.: Reinforcement learning with replacing eligibility traces. *Mach. Learn.* **22**, 123–158 (1996)
17. Stone, P., Veloso, M.: Multiagent systems: a survey from a machine learning perspective. *Auton. Robot.* **8**(3), 345–383 (2000)
18. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
19. Toader, F.A.: Production scheduling in flexible manufacturing systems: a state of the art survey. *J. Electr. Eng. Electron. Control Comput. Sci.* **3**(7), 1–6 (2017)
20. Urlings, T.: Heuristics and metaheuristics for heavily constrained hybrid flowshop problems. Ph.D. thesis (2010)
21. Vogel-Heuser, B., Lee, J., Leitao, P.: Agents enabling cyber-physical production systems. *AT-Autom.* **63**, 777–789 (2015)
22. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King’s College (1989)
23. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**, 229–256 (1992)
24. Xiang, W., Lee, H.: Ant colony intelligence in multi-agent dynamic manufacturing scheduling. *Eng. Appl. Artif. Intell.* **21**, 73–85 (2008)
25. Ng, A.Y., Jordan, M.: PEGASUS: a policy search method for large MDPs and POMDPs. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (2000)
26. Zhang, W.: Reinforcement learning for job shop scheduling. Ph.D. thesis, Oregon State University (1996)