Agent-Based Modeling and Analysis of Dynamic Slab Yard Management in a Steel Factory

Hajime Mizuyama

Agent-Based Modeling and Analysis of Dynamic Slab Yard Management in a Steel Factory

Hajime Mizuyama[1][0000-0001-9823-1832]

[1] Aoyama Gakuin University, 5-10-1 Fuchinobe, Chuoku, Sagamihara 252-5258, Japan
mizuyama@ise.aoyama.ac.jp

Abstract. The slab yard upstream of a heating furnace in a steel factory has several LIFO buffers and a crane. In managing the yard, slabs are moved from a buffer to another and to the furnace by the crane. Which specific operation should be done and when are determined by the crane operator according to the situation being dynamically changed with arrivals of new slabs to the yard and removals of heated slabs from the furnace. This study models a mathematical aspect of the dynamic decision process for managing the yard, and analyzes the effects of some factors on its performance using a reinforcement learning agent. As a result, it reveals an interaction effect between the arrival rate of slabs and how many steps in the future the agent considers when making decision. The findings obtained by this and similar following studies will be valuable for supporting the actual decision process and for training skills for the process.

Keywords: Slab Yard Management, LIFO Buffers, Reinforcement Learning.

1 Introduction

There is a heating furnace between casting and rolling processes in a steel factory, and slabs supplied from the casting process are loaded to the rolling process after being adjusted their temperature by the furnace. The yard upstream of the furnace has several stacks, i.e. LIFO buffers, of slabs and a crane, and the slabs are moved by the crane from a stack to another or to the furnace. In the factory considered in this paper, the sequence and timing of arrivals of slabs, those of loading the slabs to the rolling process are not strictly specified in advance. Hence, it becomes a dynamic decision process to determine which slab should be moved to where and when by the crane. This decision process involves quite a few objectives and constraints, such as satisfying the due date assigned to each slab, inserting slabs of similar heating conditions as consecutively as possible, ordering slabs so that the transitions of their width and thickness should fit the requirements of the rolling process. Further, their relative importance depends on the situation. In this factory therefore the dynamic decision process for managing the slab yard is carried out by the operator according to the state of the yard and the furnace, various information exchanged and negotiated with the neighboring processes, etc. Thus, it is an important issue how to effectively support

the human decision process online as well as how to efficiently train skills necessary for conducting the process offline.

Some authors assumed that the initial arrangement of slabs in the stacks is given and no further slabs will arrive afterwards, and dealt with a planning problem of crane operation for loading the slabs to the furnace and rearranging them in the stacks before loading if necessary. For example, [1-6] proposed how to minimize the number of rearrangement operations (or a certain generalized objective, such as a weighted sum of the number of rearrangement operations and the total moving distance of the crane) with satisfying a specified constraint on the loading sequence. Other authors considered both departures and arrivals of slabs. For instance, [7] minimized the number of crane movements under a given arrival schedule and a specified loading sequence. [8, 9] minimized the number of rearrangement operations under given arrival and loading sequences. [10, 11] minimized the number of crane movements under a given arrival schedule and specified loading due dates. However, all of these earlier studies capture the problem of managing the slab yard as a static optimization problem, which is essentially a different setting from the dynamic decision process to be addressed in this paper. Further, they all aim at automating and computerizing operational planning and lacked the perspective of how to support a human decision-making and how to train skills for it.

Why the task of managing the slab yard is deferred to a human in this factory is not only because it is a dynamic decision process with quite a few objectives and constraints. It is also and more essentially because various conditions and parameters necessary for formulating the decision-making problem underlying the task, such as the relative importance among the objectives and constraints, how to parameterize the uncertain future state, etc. are not fixed a priori and dependent on the changing situation, and hence the human's informal and intuitive estimate on them made on the fly in the shop floor is often helpful and can work favorably. Further, relevant information to the estimate can be attained not only by careful observation of the yard, but also by communicating and negotiating with the managers of neighboring processes. These proactive information collection and intervention can be naturally carried out by a human but are still quite difficult to automate. On the other hand, the decision-making problem itself obtained by setting the parameters and conditions with the help of fuzzy information processing by a human is still a sort of mathematical optimization problem. When considering how to support this decision-making, it will be difficult to take an approach which provides potential solutions to the human in charge, because the problem to be solved cannot be clearly captured a priori and will become complete only after being complemented by the decision maker's intuition. A promising alternative would be to provide an adequate solution framework to be taken when a human cognitively derives a solution to the problem.

Thus, this paper adopts a reinforcement learning model [12] for capturing the framework of the dynamic decision process for managing the slab yard and tries to analyze what factors affect its performance using the model. Due to the sparkling success of recent deep learning techniques and their application to reinforcement learning, (deep) reinforcement learning approaches are actively applied even to manufacturing systems. Most of such applied researches aim at enhancing and automating

dynamic scheduling and dispatching [13-24]. This paper can be distinguished against them in that it utilizes a reinforcement learning model as a framework for analyzing how a human addresses a dynamic decision making process. In the remainder, the slab yard and its management task are modelled, and then a reinforcement leaning agent is introduced. Further, numerical experiments and their results are presented, and finally the paper is concluded with potentially fruitful future research directions.

2 A Simple Model of Slab Yard and Its Management

This section provides a model of the dynamic decision process for managing the slab yard, which is considerably simplified but captures essential features of the mathematical aspect of the actual process. The outline of the model is described below.

- The heating furnace and the following rolling process are grouped into a virtual single machine for simplicity, and the slab yard is captured as a set of the machine, several buffers, and a crane.
- The buffers in the yard is classified into an entrance buffer, four intermediate buffers, and a loading buffer to the machine. Further, there is assumed to be an invisible queue upstream of the entrance buffer.
- The entrance and intermediate buffers are LIFO, and the loading buffer and the queue are FIFO. The capacity of the buffers is set to four, except that the queue has an infinite capacity.
- Slabs arrive randomly and enter into the queue. The time between consecutive arrivals follows an exponential distribution with $1/\lambda \in \{10, 11, 12\}$. Slabs in the queue are automatically moved to the entrance buffer one by one and the cycle time of this movement operation is four.
- Each slab has information on its type ($\in \{1, 2, 3, 4\}$) and due (specified by adding a random variable from a uniform distribution $[60, 720]$ to the arrival time).
- Slabs in the loading buffer are automatically loaded to the machine one by one when the machine becomes available. The processing time of each slab is six irrespective of the type.
- A setup operation is necessary before loading a slab if its type is different from that of the immediately earlier one, and its time depends heavily on whether the type is changed in an increasing direction ($= 6$) or a decreasing one ($= 60$).
- The order of processing the slabs is not rigidly specified a priori, but only loosely constrained by their due dates.
- The possible route of the crane is represented by a star graph, whose leaves correspond to the entrance, intermediate and loading buffers. The traveling time of each edge of the graph is the same.
- Every movement cycle of the crane is to start from the center node, move to a buffer, take out a slab from the buffer, travel to another buffer via the center node, release the slab there, and come back to the center node. The cycle time of this movement is four.
- A unitary bonus is given to the operator each time a slab is loaded to the machine.

- A tardiness penalty ($= \text{tardiness} \times 1/30$) is incurred for each slab which cannot be loaded to the machine before its due date.
- The objective function to be maximized is the score defined by subtracting the total tardiness penalty from the total bonus attained in a specified period of time ($= 60 \times 24$) starting from a randomly set initial state.

3 Crane Operator Reinforcement Learning Agent

3.1 Actions

The crane operator needs to decide what action to take next in every cycle time. Possible actions are to wait in the center node for a cycle or to move a slab from a buffer (origin) to another (target). When "to move" is chosen, the origin and target buffers need to be specified. Neither the loading buffer nor empty buffers cannot be chosen as the origin. Hence, the origin must be selected among nonempty entrance or intermediate buffers. Similarly, the target must be selected among non-fully occupied intermediate or loading buffers, since neither the entrance nor fully occupied buffers cannot be specified as the target.

3.2 Rewards

How much reward the crane operator receives in each cycle can be defined with the score, i.e. the value calculated by subtracting the tardiness penalty from the loading bonus. More specifically, the reward is defined by the difference of the score between before and after a corresponding action is taken.

3.3 State Features

The state of the yard changes with actions taken by the crane operator, arrivals of new slabs, and accompanied events, such as moving a slab to the entrance buffer, loading one to the machine, starting and finishing a setup operation, starting and finishing processing a slab on the machine, etc. Further, a same state may be perceived differently, and it may affect the performance of the decision process. For simplicity, this paper assumes the operator perceives the state of each buffer with the features below.

The level of the buffer, the number of slab types in it, the number of times the type changes in an increasing direction when taking out slabs one by one, that in a decreasing direction, the average due date, the number of times a consecutive pair of slabs are lined up in the order of their due dates, that in the opposite order, the number of slabs whose due date is earlier than any slab in the loading buffer, their maximum depth from the top, the number of slabs which cannot satisfy their due date without rearranging their position, their maximum depth, the maximum estimated tardiness, the depth of the slab whose estimated tardiness is the longest, the sum of the expected tardiness of all slabs, the sum of the slack times of all slabs

3.4 State Value Function

The discounted sum of the rewards which the crane operator in a specific state will achieve in the future if she/he follows a certain policy from that on is called the state value. This paper approximates the state value function by a standard multi-layer neural network whose input is the state feature vector and output is the state value.

3.5 Myopic and Forward-Looking Policies

If arrivals of new slabs are ignored, the crane operator can envision the state (or afterstate) attained as the result of taking an action from a state. Similarly, the state attained by taking another action from the afterstate, the next state attained by further taking another action, etc. can also be estimated. Thus, it is quite natural to assume that the crane operator chooses an action so that the value of an afterstate be maximized. The question is which afterstate it is. In other words, of how many steps in the future an afterstate is considered?

3.6 Learning Algorithm

This paper utilizes $TD(\lambda)$ algorithm, which learns the state value function according to so called λ return [25]. This algorithm is suitable for a dynamic decision process where afterstates can be defined as described above, and is confirmed to be effective in real-life problems such as backgammon [26].

4 Numerical Experiments

This section conducts numerical experiments using the mathematical models described above and analyzes what factors affect the performance of the decision process of managing the slab yard. Potential factors include how the state is perceived, how much discounting rate is used for calculating the state value, how many future steps are considered in each decision, etc., but this paper focuses on the effects of the number of forward-looking steps with changing the arrival rate of slabs. More specifically, experiments are conducted under nine different conditions obtained by combining three levels of the mean time between arrivals of slabs ($\in \{10, 11, 12\}$) and three levels of the number of future steps considered ($\in \{1, 2, 3\}$).

Other settings are determined according to preliminary experiments. For instance, a standard fully connected multi-layer network with two intermediate layers of 40 nodes is used as the structure of the neural network approximating the state value function. The network uses a sigmoid function as the activation function of every node. The initial value of the learning rate is set to 0.00001 and is decreased exponentially by the ratio of 0.999 in every episode. The learning process is continued up to 1000 episodes. Since the crane operator chooses an action (and a learning step is taken) for about $(60 \times 24)/4 = 360$ times in each episode, this means that the total num-

ber of learning steps is about 360,000. The discounting rate for calculating the state value is set to 0.99, and the value of parameter λ for learning is set to 0.7.

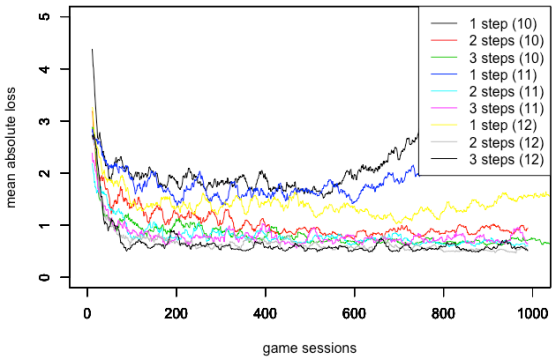

Fig. 1. Mean absolute losses.

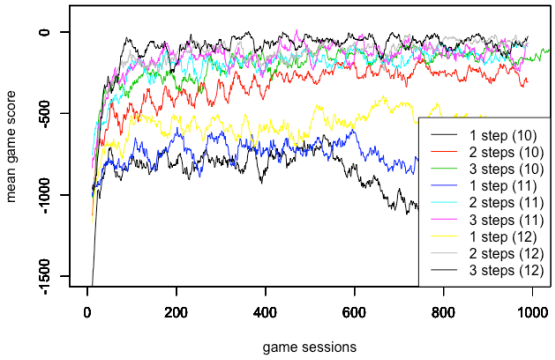

Fig. 2. Mean game scores.

Fig.1 and **Fig.2** show the results. In the figure legends, the number of future steps considered and, in the parenthesis, the mean time between arrivals (i.e. the inverse of the arrival rate) of slabs are indicated. The vertical axis of **Fig.1** shows a moving average of the mean absolute loss of each episode, that of **Fig.2** represents a moving average of the final score of each episode. A most clear result is that the smaller the loss the higher the score. It is also natural that the shorter the mean time between arrivals and the more frequently slabs arrive, the harder the problem and the lower the average score. In addition, it is observed that the average score obtained by the myopic policy with only one step forward-looking is inferior to those obtained by forward-

looking policies considering two or more steps in the future irrespective of the mean time between arrivals. Looking into further details reveals that there is no noticeable difference between the policies considering two steps and three steps when the mean time between arrivals is long but, when it is decreased to 10, the superiority of considering three steps appears. These results imply the effectiveness of forward-looking policies and that it depends on the congestion of the yard how many future steps should be taken into consideration.

5 Conclusions

This paper mathematically modelled the dynamic decision process for managing a slab yard in a steel factory and the process of how an operator learns a policy for the decision process. It then analyzed what factors affect the performance of the decision process by conducting numerical experiments using the models. As a result, forward-looking decisions are shown to be effective. Further, it depends on the congestion of the yard how many future steps should be taken into consideration.

The numerical experiments conducted in this paper focused mainly on the effects of the number of forward-looking steps considered. It would be also interesting and valuable to investigate the effects of other factors, such as how the state is perceived, how much discounting rate is used for calculating the state value. Further, it would be important to study how to utilize the findings obtained by this and following studies for supporting the actual decision process and for training skills for the process.

References

1. Tang, L., Liu, J., Rong, A., Yang, Z.: An Effective Heuristic Algorithm to Minimise Stack Shuffles in Selecting Steel Slabs from the Slab Yard for Heating and Rolling. Journal of the Operational Research Society 52, 1091-1097 (2001).
2. Tang, L., Liu, J., Rong, A., Yang, Z.: Modelling and a Genetic Algorithm Solution for the Slab Stack Shuffling Problem when Implementing Steel Rolling Schedules. International Journal of Production Research 40(7), 1583-1595 (2002).
3. Singh, K. A., Tiwari, S. M. K.: Modelling the Slab Stack Shuffling Problem in Developing Steel Rolling Schedules and its Solution Using Improved Parallel Genetic Algorithms. International Journal of Production Economics 91, 135-147 (2004).
4. Tang, L., Ren, H.: Modelling and a Segmented Dynamic Programming-Based Heuristic Approach for the Slab Stack Shuffling Problem. Computers & Operations Research 37, 368-375 (2010).
5. Cheng, X., Tang, L.: A Scatter Search Algorithm for the Slab Stack Shuffling Problem. Lecture Notes in Computer Science 6145, 382-389 (2010).
6. Tang, L., Zhao, R., Liu, J.: Models and Algorithms for Shuffling Problems in Steel Plants. Naval Research Logistics 59, 502-524 (2012).
7. Konig, F. G., Lubbecke, M., Mohring, R., Schafer, G., Spenke, I.: Solutions to Real-World Instances of PSPACE-Complete Stacking. Lecture Notes in Computer Science 4698, 729-740 (2007).

8. Kim, B.-I., Koo, J. Sambhajirao, H. P.: A Simplified Steel Plate Stacking Problem. International Journal of Production Research 49(17), 5133-5151 (2011).
9. Lu, C., Zhang, R., Liu, S.: A 0-1 Integer Programming Model and Solving Strategies for the Slab Storage Problem. International Journal of Production Research 54(8), 2366-2376 (2016).
10. Rei, R. J., Pedroso, J. P.: Heuristic Search for the Stacking Problem. International Transactions in Operational Research 19, 379–395 (2012).
11. Rei, R. J., Pedroso, J. P.: Tree Search for the Stacking Problem. Annals of Operations Research 203, 371–388 (2013).
12. Sutton, R. S., Barto, A. G.: Reinforcement Learning: An Introduction, 2nd edition. The MIT Press, Cambridge, Massachusetts (2018).
13. Wang, Y.-C., Usher, J. M.: Application of Reinforcement Learning for Agent-Based Production Scheduling. Engineering Applications of Artificial Intelligence 18, 73–82 (2005).
14. Gabel, T., Riedmiller, M.: Distributed Policy Search Reinforcement Learning for Job-Shop Scheduling Tasks. International Journal of Production Research 50, 41-61 (2012).
15. Qu, S., Wang, J., Govil, S., Leckie, J. O.: Optimized Adaptive Scheduling of a Manufacturing Process System with Multi-Skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach. Procedia CIRP 57, 55-60 (2016).
16. Shahrabi, J., Adibi, M. A., Mahootchi, M.: A Reinforcement Learning Approach to Parameter Estimation in Dynamic Job Shop Scheduling. Computers & Industrial Engineering, 110, 75-82 (2017).
17. Ou, X., Chang, Q., Arinez, J., Zou, J.: Gantry Work Cell Scheduling through Reinforcement Learning with Knowledge-Guided Reward Setting. IEEE Access 6, 14699-14709 (2018).
18. Stricker, N., Kuhnle, A., Sturm, R., Friess, S.: Reinforcement Learning for Adaptive Order Dispatching in the Semiconductor Industry. CIRP Annals - Manufacturing Technology 67, 511-514 (2018).
19. Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., Kyek, A.: Optimization of Global Production Scheduling with Deep Reinforcement Learning. Procedia CIRP 72, 1264-1269 (2018).
20. Ou, X., Chang, Q., Chakraborty, N.: Simulation Study on Reward Function of Reinforcement Learning in Gantry Work Cell Scheduling. Journal of Manufacturing Systems 50, 1-8 (2019).
21. Minguillona, F. E., Lanza, G.: Coupling of Centralized and Decentralized Scheduling for Robust Production in Agile Production Systems. Procedia CIRP 79, 385-390 (2019).
22. Kuhnle, A., Röhrig, N., Lanza G.: Autonomous Order Dispatching in the Semiconductor Industry Using Reinforcement Learning. Procedia CIRP 79, 391-396 (2019).
23. Kuhnle, A., Schäfer, L., Stricker, N., Lanza, G.: Design, Implementation and Evaluation of Reinforcement Learning for an Adaptive Order Dispatching in Job Shop Manufacturing Systems. Procedia CIRP 81, 234-239 (2019).
24. Chen, Y., Qian, Y., Yao, Y., Wu, Z., Li, R., Zhou, Y., Hu, H., Xu, Y.: Can Sophisticated Dispatching Strategy Acquired by Reinforcement Learning?: A Case Study in Dynamic Courier Dispatching System. In: Proc. of the 18th International Conference on Autonomous Agents and Multi-Agent Systems; AAMAS'19, pp. 1395-1403 (2019).
25. Tesauro, G.: Practical Issues in Temporal Difference Learning. Machine Learning 8(3), 257-277 (1992).
26. Tesauro, G.: Temporal Difference Learning and TD-Gammon. Communications of the ACM 38(3), 58-68 (1995).