A New Heuristic Reinforcement Learning for Container Relocation Problem

A New Heuristic Reinforcement Learning for Container Relocation Problem

Tiecheng Jiang[1,a], Bo Zeng[2,b], Yong Wang[1,c*], Wei Yan[1,d]

[1] College of Information Science and Engineering, Ocean University of China, Qingdao, Shandong, 266100, China

[2] Department of Industrial Engineering and Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, 15101, USA

[a]e-mail: 21180211093@stu.ouc.edu.cn, [b]e-mail: bzeng@pitt.edu, [d]e-mail: 21190231400@stu.ouc.edu.cn

[c*]Corresponding author's e-mail: wangyong@ouc.edu.can

Abstract. Many research efforts on reinforcement learning (RL) is dedicated to solving challenging combinatorial optimization problems, including the container relocation problem (CRP) arising from maritime industry. In CRP, large and heavy containers are stacked on top of each other, those above the target container should be relocated elsewhere before retrieving it. This problem, which minimizes the number of relocations in the retrieval process, is NP-hard, while existing RL studies do not provide satisfactory results yet. We study the structure of CRP problem and design a new heuristic RL framework. On a set of commonly used examples, our RL demonstrates a performance almost equivalent to the most famous results.

1. Introduction

Combinatorial optimization is a critical research subject in computer science and operations research. A very recent combinatorial optimization model is the container relocation problem (CRP), which arises from the port management and maritime logistics. In those practical scenarios, containers, which are very heavy, are piled in stacks. Clearly, to retrieve one container, i.e., the target container, all containers above it should be relocated to elsewhere. Hence, retrieving some containers from a container yard often incur many relocation operations. Note that CRP also known as BRP, is a new NP-hard combinatorial optimization problem [1].

Given a set of piled containers and their retrieval priorities, it describes a situation where the goal is to define an operation sequence with the least number of container relocation movements (Figure 1). If only the containers piled above the target one can be relocated, the CRP is called the restricted one. Otherwise, it is called the unrestricted one. Because the operational rule for the former one is much simpler, the unrestricted CRP is believed to be more challenging.

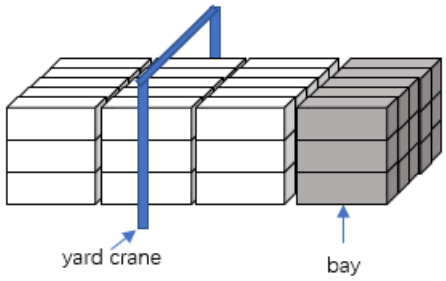

Figure 1. An overview of yard

Existing studies focus on in-depth theoretical analysis on deriving tight lower bounds on the number of relocation operations [2,3]. Also, many research efforts have been devoted to developing strong mathematical programming formula [4] to obtain the exact results, which, however, can only handle small-scale instances. The prevailing solution methods are domain-knowledge based heuristic algorithms, such as IDA * algorithm [5].

Reinforcement learning (RL) has been committed to training intelligent agents in video games and board games to achieve human like or even better performance. Among them, in the field of go, AlphaGo defeated the top players of human beings, proving that RL method has an excellent advantage in solving the goal-directed Markov decision-making problems that need to interact with the environment and obtain rewards [6]. For the traditional combinatorial optimization problem, it can be regarded as a completely-known Markov decision process, so RL has a high adaptability for combinatorial optimization problems.

It is worth mentioning that RL has been applied to solve several classical combinatorial optimization problems. To solve the travelling salesman problem, strategy gradient optimization techniques have been integrated with neural network and RL. R2 algorithm and improved MCTS algorithm have been proposed to solve the bin packing problem (BPP). An end-to-end RL framework has been proposed to solve vehicle routing problem by applying neural network, whose parameters are optimized by a policy gradient algorithm [7]. Deep RL has also be used to identify orderings by providing tight bounds, which are obtained from relaxed Decision Diagrams [8]. At the same time, researchers are also improving RL algorithm by introducing Hierarchical Reinforcement Learning to solve the problem of huge state spaces [9]. In addition, some researchers have proposed to establish static and dynamic value formulas to quantify the computations (i.e., simulated) value, thereby improving the short-sightedness of MCTS, which is assigning action and computations values.

Specifically, when using RL to calculate CRP, we notice that only using reward function cannot help the convergence of reinforcement learning algorithm. Therefore, we propose a new heuristic algorithm combined with reinforcement learning method to solve CRP. At the same time, in order to reduce the probability of falling into the local optimal solution, we make some improvements in the framework of reinforcement learning method to support the heuristic algorithm.

2. The Setup of CRP within the RL Framework

The CRP studied in this paper is an unrestricted variant. As mentioned, the state and action space of unrestricted CRP is more complex, which renders algorithms frequently fall into local optimal solutions. Following the mainstream in the literature, we consider a CRP with a single bay, and multiple stacks (Figure 2).

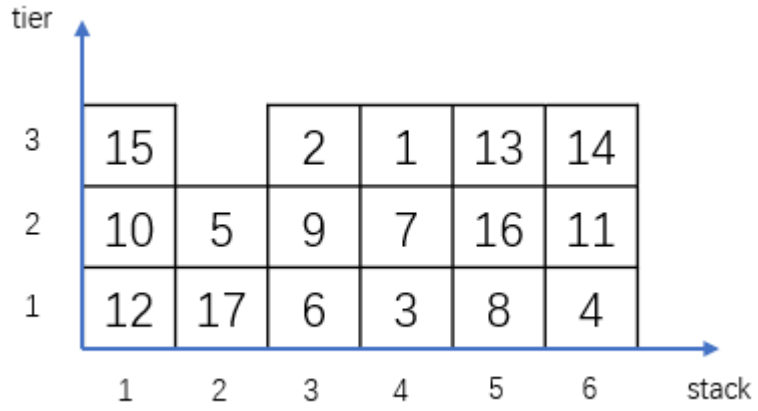

Figure 2. A CRP with a single bay, and multiple stacks

The horizontal axis keeps track of the stack number, and the vertical one labels the number of layers. Containers' IDs indicate their priorities, where the smaller ID has a higher priority. The action that moves the top container from one stack to the top of another stack is defined as a relocation operation. Moving the top container in a stack away from the current configuration is defined as a retrieval operation. When all containers are retrieved, it means that this CRP instance is solved. As mentioned, the ultimate goal of solving a CRP instance is to obtain an operation sequence with the least number of relocation operations.

In this paper, a partial modeling framework of the finite Markov decision process (MDP) is adopted. The state space is represented by s , and a state $s \in S$ is defined as the configuration of containers among stacks. The initial state s_0 is the original configuration, and the end state is the empty bay where all the containers are retrieved. According to the effect of the operation, the action space A can be partitioned into the set of retrieval operations and the set of relocation operations. Since only the top containers can be moved, we simply define action $a=(ST_1, ST_2)$, for $a \in A$ where ST_1 and ST_2 represent the original stack of the container to be moved and its destination stack, respectively. When $ST_1=ST_2$, it denotes a retrieval operation, and when $ST_1 \neq ST_2$, it represents a relocation operation. Since the environment of CRP is completely observable, the transition probability of states is 1.

In this paper, five prior constraints are considered. The prior knowledge here stipulates some legal and illegal operations that can ensure the normal change of state in reality, and does not initially give the agent some human influence:

- If there are other containers above the target container (i.e., the container with the highest priority) in the current configuration, it cannot be retrieved;
- If there is no other container above the target container, the retrieval operation must be performed;
- The layer limit of any configuration is the highest number of layers of the original configuration;
- The action that involves both the initial and destination stacks of previous relocation operation is not included in the current action space;
- If a top-level container is not the one with the highest priority, its associated actions will not be included in the current action space

3. Heuristic Reinforcement Learning

We first propose a simple and intuitive state evaluation algorithm for the unrestricted CRP(Figure 3).

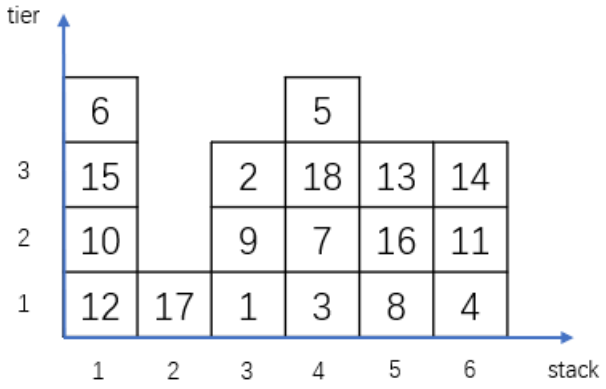

Figure 3. The blocking degrees of stack from left to right are: -5, 0, -9, -21, -13, -17; the blocking degree in this state is -65

For any stack in a configuration (i.e., a state), we identify the container with the highest priority in that stack, which separates the stack into upper and lower parts. We group the separating container into the upper part. Consider the upper part first. Clearly, if there are containers in the upper part, they must have priorities lower than the separating container. Starting from the top container in this upper part, we subtract the separating container's priority by those of each above it. For the resulting negative values, we sum them up to obtain a single value. Because of its connection to the extent of relocation efforts, we call it the blocking degree.

Next, we deal with the lower part. Again, we identify the container with the highest priority in this part, and then separate this part into upper and lower subparts with respect to that container. Repeating the operations for the upper part, we can define the block degree for the upper part. We iterate this process for the lower subpart until it can no longer be divided into two parts anymore. Finally, the blocking degree of each part is added up as the blocking degree of this whole stack. It is worth noting that the value of blocking degree for the empty stack is 0, and the blocking degree of this part is 0 if there is only a separating container in the upper half. The blocking degree of each stack is added to obtain the blocking degree of the overall state, which is used as the evaluation of the state in our reinforcement learning. In the selection and simulation stages of the algorithm, this is used as an important reference for sorting actions.

At the same time, in order to reduce the possibility of the algorithm converging to a local optimal solution, we have adjusted the framework of the reinforcement learning algorithm. The algorithm first classifies the actions, and then sorts each action group according to the blocking degree to reduce the adverse effect of the heuristic algorithm on the result. The classification criteria should be the special relocation in the problem. This paper divides the action space into two categories according to whether the initial stack forms an empty stack after the operation or whether the target stack before the operation is an empty stack. Then, each group of action is sorted according to the blocking degree to generate the action space of the algorithm. Finally, the algorithm chooses between exploration and exploitation in accordance with the principle of epsilon-greedy.

We use the above heuristic algorithm as the initial value of the value function, and use the reward function to experiment in random cases as follow:

$$reward = \begin{cases} -n \times relocationNum & if\ a\ is\ relocation \\ 1 & if\ a\ is\ retrieve \end{cases} \quad (1)$$

where *n* is the previous relocation times of the container, *relocationNum* is the total number of relocations for the current instance. At the same time, we use the TD error to update the value function.

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \quad (2)$$

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) \quad (3)$$

Formula 2 is TD error formula, where δ_t is TD error(TD error is the way to update the value function in temporal-difference learning, which is a model-free reinforcement learning method), R_{t+1} is the reward when the state is *t + 1*, $V(S_{t+1})$ and $V(S_t)$ are the value functions of *t + 1* and *t* respectively, $\gamma$is the attenuation coefficient. Formula 3 is the renewal formula of value function, where α is learning efficiency.

4. Results & Conclusions

Table 1. Our results and Comparison with best known

S-T-C	Count	Best known		New HRL	
		LB	Rel	Rel	Opt%
3-6-15	100	548	548	548	100%
3-6-16	100	678	678	679	99%
3-6-17	100	691	691	693	98%
4-6-20	100	996	996	996	100%
4-6-21	100	1149	1149	1151	98%

To validate our algorithm, we adopt a commonly used data set from Zhu [5]'s paper for computational evaluation. In Zhu et al.'s paper, authors design and implement IDA*-U algorithm, which uses IDA* as the framework and integrates fast heuristic algorithms, and generate a set of complex instances. For those instances, they perform IDA*-U algorithm on a high-performance computing server without time limit (subject to the number of nodes explored within 2^{30} to generate the best possible results on the number of relocations (*Rel*) and the expected lower bound (*LB*). Note that *LB* is not the theoretical lower bound.

In our implementation, computational results are summarized in Table 1 where *S* is the number of layers, *T* is the number of stacks, *C* is the number of the containers, *Count* is the number of instances, and *opt\%* is the proportion of instances on which our algorithm is better than or equal to the best-known results in Zhu et al.'s paper. It is worth noting that *LB* and *Rel* are the results of summing all instances. *New HRL* represents the algorithm proposed in this paper, namely heuristic reinforcement learning. From this table, we can conclude that the performance of the new heuristic reinforcement learning is almost equivalent to the best-known.

After our experiments, we believe that a heuristic that can reflect the principle of the problem can help the reinforcement learning algorithm converge, nevertheless, in order to obtain the global optimal solution, we must add the new constraints to the heuristic algorithm in the reinforcement learning algorithm framework.

References

[1] Caserta, M.; Schwarze, S.; and Voß, S. (2012) A mathematical formulation and complexity considerations for the blocks relocation problem. European Journal of Operational Research 219(1): 96–104.

[2] Tanaka, S.; and Mizuno, F. 2018. An exact algorithm for the unrestricted block relocation problem. Computers & Operations Research 95: 12–31.

[3] Lu, C.; Zeng, B.; and Liu, S. 2020. A Study on the Block Relocation Problem: Lower Bound Derivations and Strong Formulations. IEEE Transactions on Automation Science and Engineering .

[4] da Silva, M. d. M.; Toulouse, S.; and Calvo, R. W. 2018. A new effective unified model for solving the Pre-marshalling and Block Relocation Problems. European Journal of Operational Research 271(1): 40–56.

[5] Zhu, W.; Qin, H.; Lim, A.; and Zhang, H. 2012. Iterative deepening A* algorithms for the container relocation problem. IEEE Transactions on Automation Science and Engineering 9(4): 710–722.

[6] Silver, D.; Huang, A.; Maddison, J. C.; Guez, A.; Sifre, L.; Driessche, v. d. G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. Nature 484–489.

[7] Nazari, M.; Oroojlooy, A.; Snyder, L.; and Takac, M. 2018. ´ Reinforcement learning for solving the vehicle routing problem. In Advances in Neural Information Processing Systems, 9839–9849.

[8] Cappart, Q.; Goutierre, E.; Bergman, D.; and Rousseau, L.- M. 2019. Improving optimization bounds using machine learning: Decision diagrams meet deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 1443–1451.

[9] Rafati, J.; and Noelle, D. C. 2019. Learning representations in model-free hierarchical reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 10009–10010.