DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

Optimization of Container Relocation Problem via Reinforcement Learning

Optimierung des Container Relocation Problems mittels Reinforcement Learning

Lei Wei
Fuyin Wie
Sandra Schmitz
Kunal Kunal
Bernd Noche

Lehrstuhl für Transportsysteme und –logistik
Universität Duisburg-Essen

This paper presents an optimization method of Container Relocation Problem (CRP) via Reinforcement Learning (RL) based on heuristic rules. The method to calculate theoretical lowest relocation rate is also briefly explained. As the result, training models for different dimensional bays are provided. Compared to the theoretical value, the result relocation rate is acceptable with high inference speed. Furthermore, extended CRP in block will be briefly demonstrated.

[Keywords: container relocation problem; block relocation problem; reinforcement learning; ML-Agents]

In dieser Arbeit wird eine Optimierungsmethode für das Container Relocation Problem (CRP) mittels Reinforcement Learning (RL) vorgestellt, die auf heuristischen Regeln basiert. Eine Methode zur Berechnung der theoretisch niedrigsten Relocation Rate wird ebenfalls erläutert. Als Ergebnis werden Trainingsmodelle für unterschiedlich dimensionierte Bays bereitgestellt. Verglichen mit dem theoretischen Wert, ist die Relocation Rate zufriedenstellend und die Inferenz-Geschwindigkeit hoch. Außerdem wird eine erweiterte Version des CRPs die sich auf einen ganzen Containerblock bezieht, präsentiert.

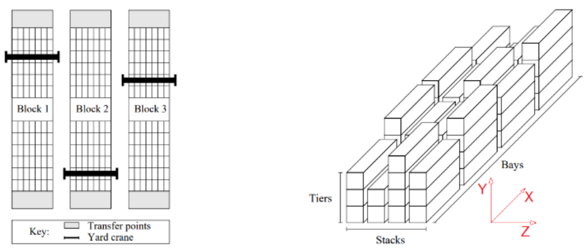
[Schlüsselwörter: Container Relocation Problem; Block Relocation Problem; Reinforcement Learning; ML-Agents]

1 INTRODUCTION

With the increase in global container trade, efficient transshipment of terminal containers is essential. Intelligent container relocation in an inland container terminal or port is significant to improve performance measures like task completion time, energy consumption, container rehandling rates and operation efficiency of a terminal.

In multimodal terminals, the cranes not only have to serve the container ships, the trucks and the railroad at the yard side, but also serve as stacking cranes. Inbound and outbound containers are often stored at the container terminals for a certain period of time, waiting to be loaded onto the train or ship, or to be delivered by trucks.

A rail-mounted gantry crane is usually used for handling containers at the terminal. The containers are stacked in storage blocks at the container yard to minimize storage space (Fig 1). Thus, only the topmost container is directly available for a retrieval. Relocations (also known as reshuffling) are necessary to grant access to a container which is not topmost of the stack. These unproductive moves performed by the yard cranes should be minimized to improve the terminal efficiency.



(a) A container yard (b) A single block

Figure 1. Container yard structure and terms (figure from [ZF12]) with coordinate directions

For this purpose, a Container Relocation Problem (CRP) (also known as Block Relocation Problem (BRP) [GBJ18]) is considered in this article.

After a brief literature review in Section 2 we will describe our approach to solve the CRP for a 2-dimensional stacking area considering one bay in Section 4. We use the ML-Agents toolkit from Unity to implement Reinforcement Learning based on heuristic rules to solve the CRP. We present the experimental results at the end of

DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

that section of our approach and a comparison to other existing approaches.

In common multimodal container terminals, the relocations won't be limited to one bay of the container blocks. The different bays of one block are not independent from each other, but can be operated by crane movements along the x-axis. We will address this issue in Section 6 by brief previewing the extension of the CRP, which aims to optimize a 3-dimensional stacking area, i.e., an entire block rather than one bay. Our main goal is to minimize the average operation (retrieval, relocation and stacking) time of containers. Besides, to visualize the process, a simulation system is developed with Unity Engine.

2 PRIOR RESEARCH

2.1 THE CONTAINER RELOCATION PROBLEM

[CVR14] classified storage yard operations in container terminals such as, storage space assignment to containers, yard crane scheduling, routing of vehicles within the terminal, optimizing relocating operations at the storage blocks, reviewing scientific journal papers between 2004 and 2012. [KE21] extended this classification by adding recent research papers.

In most of the papers for the CRP research, the objective is to find an optimal sequence of crane movement to retrieve all the containers from a bay according to a predefined retrieval sequence, so the number of movements is minimized [CSV20]. However, there are also approaches that focus on other optimization goals as minimizing the crane's working time associated with any movement like relocation or retrieval of containers [LL10], [FB12], [SAT19].

As the CRP is known as NP hard [CSV12], only small instances can be solved with exact methods in reasonable time. So, several heuristic approaches can be found in literature. For a comprehensive literature survey of the CRP and various exact and heuristic solution methods that have been applied to the CRP, we refer to [SAT19], [MGM20], [CSV20].

2.2 REINFORCEMENT LEARNING

Reinforcement Learning is one of the three basic paradigms of machine learning, together with supervised learning and unsupervised learning. Back in 1996, Kaelbling et al. described Reinforcement Learning as "the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment" [KLM96].

Proximal Policy Optimization (PPO) is a new class of reinforcement learning algorithms, which perform comparable or better performance than other modern approaches like TRPO (Trust Region Policy Optimization) while being much simpler to implement and tune [SWDRK17].

Jerry Elman [Elm93] proposed the idea of training a learning machine with a curriculum back in 1993. Bengio et al. [BLCW09] presented a summary of curriculum learning back in the day. They proposed curriculum learning as a method for a stepwise progression of the complexity of the data samples used during the training process.

In Section 4 we use PPO as the training algorithm and apply curriculum learning to accelerate the training process.

2.3 OTHER ALGORITHMS

- **Iterative Deepening A* (IDA*) algorithm** [ZQL12] [LZL20]: Zhu et al. developed IDA* algorithms for the unrestricted CRP. By using their derived dominance property, it takes advantages of two new lower bounds and several probe heuristics. Successive target containers can be retrieved as long as they are on top of their respective stacks at the time of retrieval, until the minimum equivalent layout is reached.
- **Genetic Algorithm (GA)** [MGM20] [SEE15]: Gamal et al. propose an optimization methodology for solving CRP using genetic algorithm. The computational results show the effectiveness of the proposed methodology for container terminal. It is widely applied because of its ability to locate the optimal solution in the global solution space.
- **Beam Search** [WT10]: Beam Search (BS) is a heuristic search algorithm based on breadth-first branch-and-bound algorithm. The term "beam search" was created by Raj Reddy in 1977.

3 PROBLEM DESCRIPTION

3.1 CONTAINER RELOCATION PROBLEM (CRP)

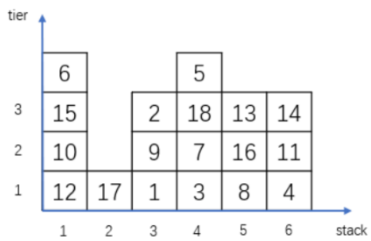


Figure 2. Layout demonstration of CRP, labels mean the retrieval priority, smaller value will be retrieved earlier [JZWW21]

DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

The goal of CRP is to minimize the relocations (or relocation rate) during the container retrieval process. Researchers use priority label to identify the retrieval sequence of containers. The container with smallest label number will be retrieved at first.

Static / Dynamic CRP: If there are no new containers during the retrieval process to be stacked on the bay, the problem is called static CRP, otherwise it's called dynamic CRP. In this paper, the static CRP will be mainly researched. Since the crane needs to serve the whole block, the dynamic CRP within one bay is usually not under consideration.

Restricted / Unrestricted CRP: CRP is restricted, if relocations are only allowed for the blocking containers above the container with highest priority. Otherwise, it's unrestricted, which means the unrestricted CRP is the super-set of restricted CRP. Generally, it has lower relocation rate and its corresponding algorithm is more complex than restricted CRP.

Stochastic CRP: If the retrieval sequence is not fully known, for instance, several containers shall be stacked on a train, then the retrieval sequence is not important as long as the corresponding containers are stacked on the correct position [BMBJ13], [GMB18].

CRP in block: In reality, relocations could happen in whole or part of container yard, which is defined as a block (Figure 1). In this scenario, the relocation rate could not be the single judgement of the problem, instead, several new judgements were introduced, like average operation time of container and average waiting time of truck [FHVX13]. Furthermore, the above-described types could be combined in this scenario.

3.2 RULES FOR UNRESTRICTED STATIC CRP

We consider the following common properties of the CRP:

- Crane performs the operation (retrieval or relocation, no stacking) with only one container at the same time.
- Only the topmost container could be operated (relocation or retrieval).
- All containers have same size.
- The relocation within the bay is limited [CVS11], which means no repeat operation is allowed.
- The operations happen only in one bay.
- The containers have unique predefined priorities, no containers have same priority.
- Bay should never be full.
- No new container will be stacked during retrieval process.
- The relocation could happen between any two stacks as long as it is possible, e.g., relocation is impossible, if target stack is full.

4 OUR APPROACH

4.1 TOOL INTRODUCING

Unity Engine: Unity Engine is a 3D real time engine for simulation and game development. For the future implementation of Digital Twins (real-time crane control), the Unity Engine was chosen to be the solution to build the host computer application.

ML-Agents: The Reinforcement Learning toolkit ML-Agents from Unity is used as training toolkit. The ML-Agents uses PPO Algorithm by default. Several learning strategies are also supported by this toolkit, such as curriculum learning, imitation learning and behavioral cloning [Uni21]. In this paper, we used curriculum learning to accelerate the training process.

4.2 RL TRAINING

In this section our training method will be introduced. The training part in this paper is designed only for relocation, since retrieval process does not need to be trained and should be determined before making any relocation decision.

The term "episode" is introduced in ML-Agents, in this context it means the period starting from initialization of new layout of the bay to finishing retrieval of all the containers in current layout. With help of this concept, the reward will be summarized during the operation process and refreshed when episode ends to ensure the rewards are for the whole episode rather than each step.

4.2.1 OBSERVATIONS OF RL

The observation structure is shown below.

Observation → Stack Info []
- (Dim-Z) Hot Encoding []
- (1) Z-index
- (1) Can pick up
- (1) Can stack
- (1) Blocking Degree
- (MaxTier * (2 + MaxTier))
- Container Info []

Container Info
- (MaxTier) Hot Encoding []
- (1) Whether moveable
- (1) Priority

Figure 3. Observation structure, each square bracket contains a list. The parenthesis means the size of the object

DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

The observation size of a stack is:

$$s_{stack} = dimZ + 4 + maxTier * (2 + maxTier)$$

Total observation size:

$$s_{total} = s_{stack} * dimZ$$

The Hot Encoding is the common way for machine learning to handle the categorical data. In this paper, the simplest One-Hot encoding was used. For corresponding z-index and tier, the element in one-hot array should be 1, other elements should be 0.

The "can pick up" property considers two aspects: (1) Whether the stack is empty. (2) Whether the stack was visited last time with unsuccessful operation. The second condition is to avoid repeating operation done by the agent, since it could fall into local optimal solution. And same with the "can stack" property, only need to change the first judgement to "whether the stack is full".

The z-index in the stack info is to ensure the trainer will get correct index after shuffling of the observations, which will help the agent not fall into local optima.

The "whether moveable" observation is to tell the trainer whether the corresponding container is moveable (relocatable). Only the topmost container could be relocated, and if the stack is empty, no container could be removed from this stack.

Jiang et al. have introduced a concept called "blocking degree" [JZWW21]. It describes how "severe" the corresponding stack is blocked. This value can be calculated by the following pseudo code:

```
define blockingDegree = 0
// elements in list are priorities
Define stack = initStack
while (stack.elementCount > 1)
    // max priority means min label
    define c = stack.MaxPriority

    // define upper stack includes c
    define hStack = stack[c.index, end]

    if (hStack.elementCount > 1)
        foreach (x in hStack exclude c)
            blockingDegree += x - c

    // update list, without c
    stack = stack[0, c.index]

return blockingDegree
```

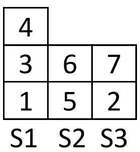


Figure 4. blocking degree calculation.

During our implementation, this concept seems to be insufficient. As Figure 4 shows, the blocking degrees of S1 and S3 calculated via the method from Jiang et al. [JZWW21] should both be 5, whereas the S1 has two containers above the container with highest priority. Thus, we introduced a new concept called "blocking count", which can be calculated similar with the "blocking degree", only needs to change the "blockingDegree += x – c" to "blockingCount += 1". The training result with the "blocking count" is slightly better than the version without "blocking count".

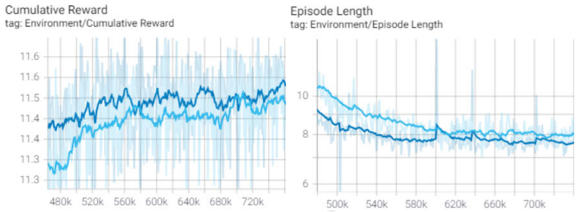


Figure 5. Comparison between with "blocking count" and without "blocking count"

4.2.2 OUTPUT

As mentioned above, all decisions the agent makes are for relocation, retrieval will be automatically determined before requesting decision from agent brain.

Action (output) can be described as (z_0, z_1), z_0 means the pick-up index, z_1 is stack index. Obviously, z_0 and z_1 should have different value, besides, stack of z_0 must not be empty and stack of z_1 must not be full. If all the containers in the bay are retrieved, the episode of current scenario is finished, a new episode will begin to continue training until it reaches the predefined max step.

4.2.3 REWARDING SYSTEM

- **Minus "repeat times", if the agent performs a repeat operation**. A repeat operation means the action is same with the last one when the last operation failed. Without this punishment, the agent will keep repeating unsuccessful action. To ensure this rule will be well followed, the reward value is not normalized.
- **Minus "0.1" every step.** The more step the agent takes to retrieve all the containers, the more punishment it will get.
- **Add "1" if a container is retrieved.**
- (optional) **Minus "0.01 * z_1".** This will encourage the agent to relocate the container near the waiting position of truck.

DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

4.3 THEORETICAL OPTIMAL SOLUTION

The theoretical optimal relocation rate can be archived via tree-search, by which all the possibilities of relocation in a layout could be achieved. Repeat of operations should be avoided during implementation, otherwise the program will fall into infinity loop. Our solution to solve the repeating problem is check of all parent nodes to see whether there are nodes which have same layout with current node.

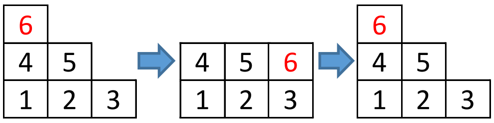


Figure 6. example of repeat operation case

There are other methods to accelerate the traverse process, e.g., ignoring all the meaningless relocation and using thread to fully use the power of CPU to run instances simultaneously, etc.

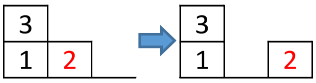
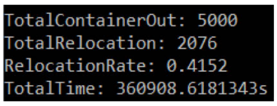


Figure 7. example of meaningless relocation

Although, the traverse process is still very slow. Figure 8 shows the result of layout 4 * 3. The Total Time means the sum of time span that each instance took, average calculation time for one layout is 12 min. More detailed heuristic will accelerate the process remarkably. Another important point is that the traverse algorithm has no memory. For two scenarios with exactly the same layout, it will cost exactly double time. Despite this method can obtain the theoretical min relocations, it's not practical for reality usage.



```
TotalContainerOut: 5000
TotalRelocation: 2076
RelocationRate: 0.4152
TotalTime: 360908.6181343s
```

*Figure 8. Theoretical lowest relocation rate for 4 * 3 layout*

The corresponding code can be found under https://github.com/idea-lei/CRP_LowBound.

4.4 EVALUATION

Max label: The initial container amount in the bay [WT10], which ensures no relocation are blocked.

$$maxLabel = (stack - 1) * maxTier + 1$$

Average relocation rate: how many relocations are needed to move one container out.

$$ART = \frac{\sum relocationTimes}{n_{container}}$$

Optimization ratio*: positive value means better result than best known.*

$$Opt = \frac{Rel_{bestKnown} - Rel_{author}}{Rel_{best\ known}} * 100\%$$

The result of each scenario from this paper has at least 1000 instances (scenarios) to reduce fluctuation of the value, and the layouts are fully random generated. Although large number of instances were tested to reduce the fluctuation of the result, there still can be around 5% error that can't be eliminated due to different layouts. Compared with the theoretical optimal relocation rate, the trained model will infer the result within 0.1s, which is much faster than the tree search.

Average relocations (relocation rate), optimization ratio (opt)							
stacks * tiers	No of containers	Theoretical optimal relocation rate	Livia Maglic (2019) - GA	Jovanovic, Voß (2014) – chain F	Wu, Ting (2010) - BS, B&B	Authors - RL	opt %
3 * 3	7	3.01 (0.430)	3.38 (0.482)	3.38 (0.482)	**3.38 (0.482)**	3.20 (0.457)	5.19%
3 * 4	9	5.06 (0.562)	5.85 (0.650)	5.95 (0.661)	**5.67 (0.630)**	5.71 (0.635)	-0.79%
4 * 3	10	4.15 (0.415)	4.98 (0.498)	4.95 (0.495)	**4.85 (0.485)**	4.51 (0.451)	7.01%
4 * 4	13	-	8.55 (0.658)	8.57 (0.659)	**8.43 (0.648)**	8.62 (0.652)	-0.60%
5 * 3	13	-	5.80 (0.446)	5.80 (0.446)	**5.75 (0.442)**	5.77 (0.444)	-0.45%

Table 1. Comparison of results with different methods from other authors. The theoretical optimal solution is not fully listed because of the time consumption.

DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

Code and test results for this section (unrestricted static CRP) can be found under https://github.com/idea-lei/CRP.

5 CONCLUSION

The static unrestricted CRP within one bay using RL was discussed in this paper, the training result is acceptable compared with theoretical lowest relocation rate but with much more time efficiency. The training is suitable for small size layout, for large layout, the training time will be relatively longer. The disadvantage of the method is that training relies much on experience to adjust the parameters of trainer, such as learning rate and hidden layers. Different configurations could lead to different result. Furthermore, the current version of ML-Agents toolkit (Release 18) could have bug, sometimes the training process could fail without any sign. We have needed to dynamically change the learning rate to keep the training process not to fail, and if the learning rate is too low, the model cannot be trained.

Figure 9. Example of training failure

6 FUTURE WORK

The further work is separated in two parts, the first is to extend the problem definition, where the CRP should be combined with the crane scheduling problem (CSP), since CRP within one bay is not practical to be used in terminals. The second is to implement the Digital Twins for the terminal.

6.1 DYNAMIC RESTRICTED CRP IN BLOCK COMBINED WITH CRANE SCHEDULING PROBLEM

In reality, the dynamic restricted CRP shall be considered in block (or whole container yard). The unrestricted CRP in block won't be considered, because the priorities of the containers could change dynamically due to the actual truck arrival time and stacking of new containers. Besides, the static CRP in whole block won't happen often in practice, so it won't be a mainstream topic, neither.

The judgement of the CRP&CSP in block can vary, Fotuhi et al. [FHVX13] have introduced a method to reduce the average truck waiting time (despite it only considered about the CSP). This is a view for truck drivers. We intend to optimize the crane operation time of each container, which will maximize the port efficiency. These two judgements are almost the same and will collapse to be exactly the same if every truck takes only one container.

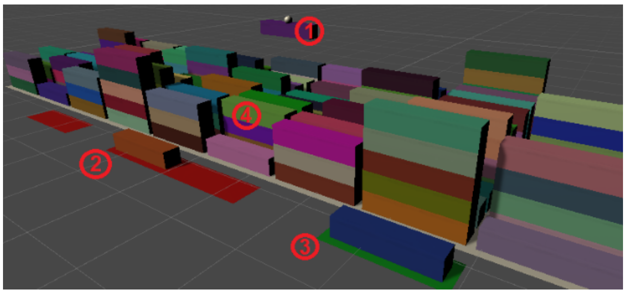


Figure 10. Layout of the simulation, 1) crane hooks the picked-up container, 2) retrieval area in red, 3) stacking transporter in green, 4) the current block of this crane

6.2 DIGITAL TWINS

The approach is also intended for future usage like Digital Twins for real-time control of cranes. Much more visualized data will be granted with the development of the concept "Digital Twins". Which means the management could become more intuitionistic with visualization of the operation process of the crane. The biggest challenge is to obtain the position data from all the moving objects (trucks, crane, etc.) and control of the objects.

7 ACKNOWLEDGEMENTS

Parts of the work were carried out as a part of the BMBF research project 01IS19068A "KI-LiveS" (KI-Labor für verteilte und eingebettete Systeme) and supported by Federal Ministry of Education and Research (BMBF).

LITERATURE

[BLCW09] Bengio, Yoshua; Louradour, Jérôme; Collobert, Ronan; Weston, Jason: *Curriculum learning*, ICML '09: Proceedings of the 26th Annual, International Conference on Machine Learning, 41–48, 2009.

[BMBJ13] Borjian, Setareh; Manshadi, Vahideh H.; Barnhart, Cynthia; Jaillet, Patrick: *Dynamic Stochastic Optimization of Relocations in Container Terminals*. Working paper, MIT, 2013.

[CSV12] Caserta, Marco; Schwarze, Silvia; Voß, Stefan: *A mathematical formulation and complexity considerations for the blocks relocation problem.* European Journal of

DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

Operational Research, Volume 219, Issue 1: 96-104, 2012.

[CSV20] Caserta, Marco; Schwarze, Silvia; Voß, Stefan: *Container Rehandling at Maritime Container Terminals: A Literature Update*. In: Böse J.W. (eds) Handbook of Terminal Planning. Operations Research/Computer Science Interfaces Series. Springer, Cham, 2020

[CVR14] Carlo, Héctor J.; Vis, Iris F. A.; Roodbergen, Kees Jan: *Transport operations in container terminals: Literature overview, trends, research directions and classification scheme*. European Journal of Operational Research, Volume 236, Issue 1: 1-13, 2014.

[CVS11] Caserta, Marco; Voß, Stefan; Sniedovich, Moshe: *Applying the corridor method to a blocks relocation problem. O*R Spectrum, Volume 33: 915-929, 2011.

[Elm93] Elman, Jerry: *Learning and development in neural networks: The importance of starting small.* Cognition, 48:781-799, 1993.

[FB12] Forster Florian; Bortfeldt Andreas: *A tree search heuristic for the container retrieval problem*. In: Klatte D., Lüthi HJ., Schmedders K. (eds) Operations Research Proceedings 2011. Springer, Berlin, Heidelberg, 2012.

[FHVX13] Fotuhi, Fateme; Huynh, Nathan; Vidal, Jose M.; Xie, Yuanchang: *Modeling yard crane operators as reinforcement learning agents*, Research in Transportation Economics, Volume 42, Issue 1: 3-12, 2013.

[GBJ18] Galle, Virgile; Barnhart, Cynthia, Jaillet, Patrick: *A new binary formulation of the restricted Container Relocation Problem based on a binary encoding of configurations*, European Journal of Operational Research, Volume 267, Issue 2: 467-477, 2018.

[GMB18] V. Galle, V. H.; Manshadi, S.; Borjian Boroujeni, C.; Barnhart, P. Jaillet: *The Stochastic Container Relocation Problem.* Transportation Science Volume 52, Issue 5, 2018.

[JV14] Jovanovic, Raka; Voß, Stefan: *A chain heuristic for the blocks relocation problem.* Computers & Industrial Engineering, 75: 79–86, 2014

[JZWW21] Jiang, Tiecheng; Zeng, Bo; Wang, Yong; Wei, Yan: *A New Heuristic Reinforcement Learning for Container Relocation Problem.* Journal of Physics, Conference Series, 1873 012050, IWECAI, 2021

[KE21] Kizilay, Damla; Eliiyi, Deniz Türsel; *A comprehensive review of quay crane scheduling, yard operations and integrations thereof in container terminals*. Flexible Services and Manufacturing Journal, Volume 33: 1-42, 2021.

[KLM96] Kaelbling, Leslie P.; Littman, Michael L.; Moore, Andrew W. *Reinforcement Learning: A Survey.* Journal of Artificial Intelligence Research, 4: 237–285.

[LL10] Lee, Yusin; Lee, Yen-Ju: *A heuristic for retrieving containers from a yard,* Computers & Operations Research, Volume 37, Issue 6: 1139-1147, 2010.

[LZL20] Lu, Chao; Zeng, Bo; Liu, Shixin: *A study on the block relocation problem: Lower bound derivations and strong formulations.* IEEE Transactions on Automation Science and Engineering, Volume 17, Issue 4: 1829-1853, 2020.

[MGM20] Magli, Livia; Gulic, Marko; Maglic, Lovro: *Optimization of container relocation operations in port container terminals*, Transport, Volume 35, Issue 1: 37-47, 2020.

[SAT19] da Silva Firmino, Andresson; de Abreu Silva, Ricardo Martins; Times, Valeria Cesario: *A reactive GRASP metaheuristic for the container retrieval problem to reduce crane's working time.* Journal of Heuristics, Volume 25, Issue 2: 141–173, 2019.

[SEE15] Said, Gamal Abd El-Nassar A.; El-Horbaty, El-Sayed M.: *An optimization methodology for container handling using genetic algorithm.* Procedia Computer Science, 65: 662-671, 2015.

[SWDRK17] Schulman, John; Wolski, Filip; Dhariwal, Prafulla; Radford, Alec; Klimov, Oleg: *Proximal Policy Optimization Algorithms*, eprint arXiv1707.06347, 2017.

[Uni21] Unity-Technologies: Training ML-Agents, release 18, 2021. https://github.com/Unity-Technologies/ml-agents/blob/release_18_docs/docs/Training-ML-Agents.md

DOI: 10.2195/lj_Proc_wei_en_202112_02
URN: urn:nbn:de:0009-14-54466

[WT10] Wu, Kun-Chih; Ting, Ching-Jung: *A Beam Search Algorithm for minimizing reshuffle operations at container yards*. Proceedings of the International Conference on Logistics and Maritime Systems, 2010.

[ZF12] Zehendner, Elisabeth; Feillet, Dominique: *Column Generation for the Container Relocation Problem.* 12th IMHRC Proceedings, 2012.

[ZQL12] Zhu, Wenbin; Qin, Hu; Lim, Andrew; Zhang, Huidong: *Iterative* deepening *A* algorithms for the container relocation problem*. IEEE Transactions on Automation Science and Engineering, Volume 9, Issue 4: 710-722, 2012.

Lei Wei, M.Sc., Researcher at the Chair of Transport Systems and Logistics, University Duisburg-Essen. E-Mail: lei.wei@uni-due.de

Fuyin Wei, M.Sc., Researcher at the Chair of Transport Systems and Logistics, University Duisburg-Essen. E-Mail: fuyin.wei@uni-due.de

Dipl.-Ök. Sandra Schmitz, Researcher at the Chair of Transport Systems and Logistics, University Duisburg-Essen. E-Mail: sandra.schmitz@uni-due.de

Kunal, Kunal, B.Sc., Research Assistant at the Chair of Transport Systems and Logistics, University Duisburg-Essen. E-Mail: kunal.kunal@stud.uni-due.de

Address: Chair of Transport Systems and Logistics (TuL), University Duisburg-Essen, Keetmanstr. 3-9, 47058 Duisburg, Germany.