Optimization of Container Operations at Inland Intermodal Terminals

Chiara Colombaroni, Gaetano Fusco, Natalia Isaenko
Dipartimento di Ingegneria Civile Edile e Ambientale
University of Rome "Sapienza"
Rome, Italy
chiara.colombaroni@uniroma1.it

Luca Quadrifoglio
Zachry Department of Civil Engineering
Texas A&M University,
College Station, Texas, USA

***Abstract*— The paper deals with the problem of minimizing reshuffling of containers in an inland intermodal terminal. The problem is tackled according to a simulation-optimization approach. A simulation model computes the operational costs of containers, related to storage and pick-up operations in an inland yard. The optimization is carried out by two genetic algorithms that work in series. The introduction of the second genetic algorithm and the concept of trust region are the original contributions of the paper to the literature. The proposed optimization method has been tested on a theoretical example of realistic size. Results highlighted that the double genetic algorithm reduces the total operational costs by 7% with respect to the single genetic algorithm.**

Keywords—Inland Freight Terminal Optimization, Reshuffling, Double Genetic Algorithm, Simulation

I. INTRODUCTION

Containerization is an efficient way to transport many kinds of goods, as it allows stocking and transporting them in the same container without handling them individually; it also reduces paperwork and damages to cargos, and speeds up transfers between modes of transport.

Many operations can be fully automated, especially at maritime ports, where standard containers are handled and large freight volumes make capital investments profitable.

Different conditions hold at inland terminals, where most operations are carried out by human operators with little help from automation, because of many non-standard intermodal transport units to handle (inland containers and swap bodies) and smaller freight volumes to shift between smaller inland vehicles (trains and trucks), as opposed to maritime transport (trains and vessel or different vessels). Moreover, because of the generally reduced availability of space in the inland yards, long and cumbersome reshuffling of containers may be required before retrieving a set of containers that are to be loaded on a train. In general, two types of container retrieval operations can be identified [1]:

- a productive move: when a container is moved directly from its storage location to a truck or train waiting to pick it up;
- an unproductive or reshuffling move: when it is necessary to pick up a container to retrieve another one stored underneath it in the same stack.

Unproductive moves heavily affect operational costs of the inland terminal because of fuel consumption and wear of both crane tires and yard pavement, as well as because of the time spent to perform them.

However, advances in container identification technology, real-time management of dynamic databases, accurate positioning techniques and short distance communications would allow today to implement real-time updated techniques to efficiently assist the operators in handling containers and minimize unproductive moves.

An extensive literature has been produced in the last two decades on container relocations at maritime terminals. This is a NP-hard problem, which has been tackled with different methods: exact solutions, heuristic algorithms and simulation models. A wide review on the specific problem of container reshuffling was made in 2011 by Caserta et al. [2] and more recently in 2015 by Tang et al. [3]. A recent literature on the more general problem of storage yard operations has been analyzed by Carlo et al. [4], who divided the container reshuffling issues into four main types; consisting of:

- minimizing reshuffling, which means that storage locations for arriving containers are selected considering expectations on their future retrieval sequence;
- performing joint retrieval sequencing and reshuffling to meet a predefined loading plan and its corresponding container retrieval sequence;
- pre-marshaling, that is the operation to reorganize the container stacking beforehand, such that for a known retrieval sequence no reshuffling will be required.
- re-marshaling, that is the operation to remove containers from their current storage locations and place them in a separate area in the yard assigned to a specific vessel.

The problem tackled in this paper belongs to the first stream.

In this context, Yang and Kim [5] studied the problem of selecting the storage locations and proposed a simple Integer Programming (IP) formulation and a GA-based heuristic for the static version of the problem, under the assumption that the containers are classified into groups.

In [6], Wan et al. (2009) dealt with the static problem to empty a given stack without any new container arrival and proposed an optimum reshuffle sequence identified by an integer program based on a new heuristic and three existing heuristics: Lowest Slot (LS) heuristic (i.e., select the open storage location in the lowest tier), the Reshuffling Index (RI) heuristic of Murty et al. [1], and the Expected Number of Additional Relocation heuristic from Kim and Hong [7]. Four constructive heuristics were proposed. The conclusion was that a modification on the rule sequence of the method from Caserta et al. [8] provided the best performances. Without providing specific details, the authors claimed that with a slight modification their solution method outperformed the one by Lee and Lee [9].

Several authors studied the impact of having more detailed information in tackling the first reshuffling decision. Kang et al. considered container storage strategies for same-weight group stacking of export containers underweight-information uncertainty [10]. A formula to estimate the number of re-handlings under uncertain weight information and some ordering constraints was derived. A stacking strategy based on Simulated Annealing algorithm, which takes into account the probability distribution of the true weight group, was developed and shown to reduce the number of re-handles compared to the traditional strategy. Zhao and Goodchild simulated the effect of considering container departure time information to minimize the number of reshuffles and intra-bay gantry crane travel distances applying three different heuristics [11]. Like Borgman et al. in [12], they concluded that a complete arrival sequence is not required to substantially reduce the number of reshuffles. As expected, the number of reshuffles decreases as the storage capacity of the bay increases. In [13], Van Asperen et al. used the models of Dekker et al. [14] and Borgman et al. [12] to evaluate the impact of truck departure time information on the time to retrieve a container, the stacking cranes workload, and the number of reshuffles required. Experimental results showed that truck announcement information has a significantly positive impact on the exit time of containers. However, announcements made with too much anticipation could be counterproductive in their implementation because more reshuffles might be incurred with newly stored containers in order to keep a container unblocked. Truck announcement information was found particularly beneficial for environments where there is unreliable or no information.

Very few authors dealt with handling of Intermodal Transport Units (ITU) in inland terminals (with few, where the problem is complicated because of the different types of ITU and different types of cranes (quay cranes, reach-stackers, forklifts). Noticeable exceptions are [15] and [16].

In [15], Carrese and Tatarelli determined a feasible sequence of stacking operations that minimize the costs associated to the storage of the ITU considering spatial and operative constraints. They proposed to solve a combined problem of container reshuffling and path optimization: a heuristic solving procedure based on a double string genetic algorithm integrated by an algorithm able to estimate the number of reshuffles.

In [16], Pap et al. analyzed the crane scheduling for reloading of containers from/to trains at inland intermodal terminals and developed a branch and bound method.

In this paper, we deal with the optimization of reshuffling in an inland intermodal terminal as a part of a more general dynamic procedure that solves the optimization problem of storage and pick-up operations in an inland yard.

With respect to existing algorithms in the literature, we introduce here specific constraints to take into account some physical issues that restrain the possible solutions, such as container resistance, and we introduce the concept of trust region with the aim of reducing the search region where a blocking container can be reshuffled.

II. METHODOLOGY

The problem is tackled according to a simulation-optimization approach. The simulation model computes the operational costs of containers associated with storage and pick-up operations in an inland yard. Optimization is carried out by two genetic algorithms applied in series every time a train arrives at the terminal and determine the optimal positions of the ITUs to unload.

The first genetic algorithm finds a sub-optimal location to storage the containers arrived at the terminal with the aim of minimizing the number of actual and expected moves. The objective function also considers both the costs for reshuffling the blocking containers and the expected costs for the future retrievals of the containers depending on their expected departure time.

The sub-optimal solution for the relocation of such blocking containers gives rise to a second optimization problem; that is: where to relocate the blocking containers that have been moved. The second genetic algorithm searches for a set of optimal positions to storage both the arrived and the blocking containers with the aim of minimizing the number of overall future retrieval movements.

Genetic algorithms are stochastic heuristic methods that demonstrated to be quite effective to solve non-convex discrete optimization problems. However, they also exhibit poor convergence properties and become inefficient for large size problems. To reduce the size of the sub-problem of reshuffling each single ITU, the space of possible solutions to investigate is reduced to a ‘trust region’ near the container to stack, since it is reasonable to assume that very far new positions would unlikely be convenient.

The logic of the algorithm is depicted in the flow chart of Figure 1. It is assumed that the operations of containers loading (from the storage area to the trains) and unloading (from the train to the storage area) do not occur simultaneously. Thus, the model simulates the arrival of one train at a time at the inland terminal.

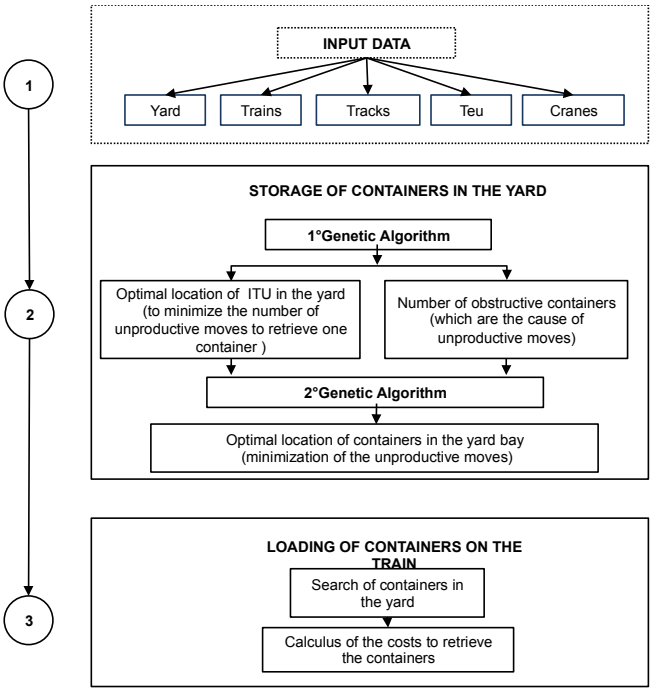

Fig. 1. Flow chart of the solution method.

The problem is tackled by assuming the following inputs:

- layout of the container yard, described by the number of stacks, tiers and rows;
- characteristics of containers: length, resistance, weight, expected departure time;
- number and characteristics of cranes (during each operation of containers loading and discharging is used only one crane);
- the list of containers to retrieve.

The simulation-optimization solution method carries out the following steps:

1. Read the list of containers to retrieve; for each container, identify the blocking ITU, if any, and the related trust region;
2. Check the constraints compliance and widen the trust region, if necessary;
3. Apply the optimization model based on expected departure time and height of stack;
4. Compare the different solutions obtained at step 3 and choose the one that minimizes the number of moves;
5. Update the list and go back to step 1 until the list of ITUs to retrieve is empty.

At the end of each simulation step the bay layout is updated. Operations of train unloading are optimized by the implementation of genetic algorithms; however, during the operations of train loading only the costs to retrieve the containers are computed without the need of applying the optimization algorithm.

III. FORMULATION PROBLEM

The problem consists of the minimization of the costs related to unproductive moves triggered by retrieving Intermodal Transport Units from a storage yard. The following hypotheses are introduced:

- all containers at the yard-bay have the same size;
- only one crane is used to handle the ITUs;
- the initial configuration of the container bay and the expected time to retrieve each ITU are known in advance.

The following notations are used to describe the problem:

- U is the number of ITU to storage in the yard bay;
- F is the number of stacks in the yard bay;
- R is the number of rows in the yard bay;
- L is the number of levels in the yard bay;
- I and are J the sets of nodes that define a complete graph and represent, respectively: the positions of ITU on the trains and all the possible positions in the yard bay.

The objective function (or fitness function in genetic algorithm jargon) entails the total operating costs associated with storage operations, i.e. the costs required to storage and reshuffling the ITUs, determined according to a feasible sequence of stacking operations. It is formulated as a linear combination of the total costs of the reshuffles and the total cost of crane movements:

$$C = Min\left(\sum_{u=1}^{U}\sum_{f=1}^{F}\sum_{r=1}^{R}\sum_{l=1}^{L} c_{frl}^{u} x_{frl}^{u} + \sum_{i\epsilon I}\sum_{j\epsilon J} g_{ij} x_{ij}\right) \quad (1)$$

where:

c_{frl}^{u} represents the unit cost of one reshuffle carried out by a crane to stack the ITU u in the cell identified by line f, row r and level l;

g_{ij} represents the unit movement cost of a crane to pick up and deliver one ITU in the storage bay;

x_{ij} represents the link between nodes I and J of the graph in the inland terminal;

x_{frl}^{u} and x_{ij} are decision variables that can only take values 0 or 1, and specifically:

x_{frl}^{u}= 1 if the ITU u is assigned to the cell identified by line f, row r and level l;

x_{ij}=1 if the link ij if the crane task is to move from node i to node j.

The method not only computes the actual costs of stacking an ITU u if the assigned position is currently obstructed by a blocking ITU, but aims at predicting also the future moves, if any, needed to reshuffle the ITU u if it blocks an ITU that is expected to be picked up before it. Of course, the current expected cost of ITUs unloading can be modified by other successive trains to unload. According to Carrese and Tatarelli [15], the reshuffling costs c_{frl}^{u} is composed of four possible terms, specified in the following paragraphs.

$c_{frl}^{u,1}$ is the unit cost of a reshuffle that is expected to incur in the future to retrieve an ITU, if any, that is stored underneath the position *f*, *r*, *l* assigned to ITU *u* and has to be retrieved before *u*;

$c_{frl}^{u,2}$ is the unit cost of a reshuffle needed to stack the ITU *u* in the assigned cell *f*, *r*, *l*, which is obstructed by an ITU, if any, stacked above cell *f*, *r*, *l*;

$c_{frl}^{u,3}$ is the unit cost of a reshuffle that is expected to incur in the future to retrieve an ITU, if any, that has to be retrieved before the ITU *u* and is stacked at the same level *l* of a position *f*, *r*, *l* assigned to the ITU *u*, which cannot be directly reached by the crane;

$c_{frl}^{u,4}$ is the unit cost of a reshuffle that is expected to incur in the future at the retrieval of the ITU *u* if it is obstructed by an ITUs that has to be retrieved after *u* and then will have to be removed to retrieve *u*.

It is worth noting that $c_{frl}^{u,2}$ is the only component of the reshuffling cost incurs to stack the ITU *u*. The other expected cost components are associated with future operations, performed to either retrieve the ITU *u* obstructed by a blocking ITU ($c_{frl}^{u,4}$) or retrieve a blocking ITU underneath ($c_{frl}^{u,1}$) or adjacent to ($c_{frl}^{u,3}$) the ITU *u*. Note also that the components of reshuffling costs due to adjacent blocking ITUs do not occur in the case of gantry cranes and have been introduced to take into account the case of reach stacker cranes.

A. Constraints

The constraints associated with the problem are defined here below. Equation (2) defines the Boolean binary variable x_{frl}^{u}:

$$x_{frl}^{u} \in \{0,1\} \tag{2}$$

Equation (3) imposes that only one ITU can be stored in a yard bay position x_{frl}:

$$x_{frl}^{u} + x_{frl}^{u'} = 1 \; \forall \, u \neq u'; u, u' \in U \tag{3}$$

$$x_{frl}^{u} + x_{frl}^{u'} = 1 \; \forall \, u \neq u'$$

Equation (4) prevents empty slots underneath the ITU *u*:

$$if \; x_{frl}^{u} = 1 \implies \sum_{h=1}^{l-1} x_{frh} = l - 1 \tag{4}$$

Equation (5) requires the weight p^u of the ITU *u* not to exceed the tightest limit of structural strength *k* of the underneath UTIs.

$$p^u x_{frl} \leq \min_h \{k_h x_{frh}\}, h = 1,2 \ldots, l-1 \tag{5}$$

IV. Solving procedure

The solution framework is based on two genetic algorithms that combine the feasible set of positions at yard bay where to store the ITUs with the aim of minimizing the objective function for the ITUs unloaded from a train arrived at the yard (the first genetic algorithm) and the ITUs to reshuffle (the second genetic algorithm). The solution method implements the set of constraints to identify the feasible solutions and compute the number of necessary reshuffles to stack the ITU to unload as well as to retrieve it or the reshuffled ITUs in the future. The solution procedure is applied every time a new train arrives and has to be unloaded. So, the problem is formulated as a dynamic process. The two genetic algorithms have the same structure, described in the following.

A. Genome definition

The genetic coding (or genome) is a straightforward representation {*f*, *r*, *l*} of the cells in the yard bay, to which all identifying codes of the ITUs are associated. Any possible genome makeup defines an individual in the population and identifies a possible solution of the ITU storage problem. Any time a new train arrives and is unloaded or leaves the terminal and is loaded, the yard bay configuration is changed and the genetic code is modified accordingly.

B. Initial population

The first step of the genetic algorithm is to generate the initial population, formed by a predefined number of possible solutions, each of which is characterized by a different genome makeup. The initial population is generated by a random procedure that ensures each ITU be assigned to a position complying with the constraints.

C. Cross-over

Cross-over operator combines the genetic patrimony of a pair of individuals (parents) to generate a pair of new individuals (children). The cross-over rule applied to generate the genome makeup of the children chooses randomly a crossover point for each pair of parents; one child receives all genes before this point from the first parent and all genes after the crossover point from the second parent; the second child receives the complementary genes. The new individuals, before being introduced in the population, will undergo a verification aimed at avoiding unfeasible solutions.

D. Mutation

A mutation operator applies, with a probability γ, random alterations to one or more genes of any individual of the population and assign a new feasible cell in the yard bay to the corresponding ITUs. Through the mutation, the algorithm can so explore new areas in the solution space. To prevent the algorithm being trapped into local minima, after a given number of iterations without any improvement of the objective function, the mutation probability γ is changed linearly until a threshold value γ_{max}. As an improvement of the objective function is obtained, the mutation probability is reset to its initial value γ.

E. Elitism

Elitism feature consists of preserving a fraction η of the best solutions for the new generation.

F. Fitness evaluation

Fitness coincides with the objective function. To evaluate the fitness, the reshuffles costs and the crane paths are computed. The first value is obtained by applying the function (1) subject to the constraints (2)-(4), accounting the effects of the dimension

and retrieval time for the ITUs and the limits on the climbing power for the cranes.

G. Algorithm application

The two genetic algorithms are applied sequentially to deal with the ITUs unloading and reshuffling in the first stage of optimization.

The optimal solution of the first genetic algorithm is represented by the minimum costs needed to reshuffle the ITUs already in the bay that obstruct the direct storage of the ITUs unloaded in the position assigned to them in the bay.

The peculiarity of the procedure lies in the second step of the optimization, which is run after the implementation of the first genetic algorithm has provided its solution to optimize also the storage positions of the ITUs to reshuffle.

V. Application to a Test Case

A. Assumptions

The optimization procedure has been applied in a simulated inland terminal where the handling operations of containers are carried out by reach stackers. In the simulation, all ITUs arrive at the terminal and depart from it by rail. This assumption simplifies the implementation of the simulation without affecting the nature of the problem, since it does not alter the processing of the ITUs, which are bunched to be loaded onto a train. It is assumed that the arrival times of all trains and the departure times of all ITUs are known in advance.

The inland terminal is composed by a 3-level yard bay having a capacity of 999 ITUs, one rail where trains arrive and depart, and an operational area where the reach stackers move from the train to the bay and *vice versa*. The yard bay track is aligned with the first line of the bay, from which the reach stacker operates to stack or retrieve the ITUs.

The yard bay has 999 possible positions that are accessible only from one side and allow ITUs being stacked up to 3 levels. The 40% of positions are initially occupied.

Each train carries 60 20-feet long ITUs. Arriving and departing trains alternate at the terminal. The time interval between successive trains is long enough to allow loading or unloading operations not to affect each other.

Since the solution procedure aims at minimizing the costs for ITUs reshuffles at the terminal by predicting the future retrievals, testing its effectiveness requires simulating the entire process of storage and retrieval all the ITUs handled. The test provides 15 simulations of the entire storage-retrieval process of the bay. Each simulation run includes 7 arriving alternated with 7 departing trains and followed by 8 departing trains that empty the yard bay. In the simulation, actual costs instead of predicted ones are computed.

In the test, the double genetic algorithm is compared with the single genetic algorithm, which only optimizes the positions of the unloaded ITUs by minimizing the number of expected reshuffles. In other words, the number of actual and expected reshuffles is minimized but their repositioning is not optimized.

B. Results

The trend of costs actually incurred for ITUs handling during the unloading phases for both the double and the single genetic algorithms are shown in Fig.2. As expected, the double genetic algorithm has a higher cost than the single one (specifically, by 8% on average) because its unloading phase includes also the repositioning of blocking ITUs in the desired cells. As shown in Fig.3, the higher costs incurred in the unloading phases by the double genetic algorithm are more than compensated in the loading phases, because the ITUs to load have been already placed in suitable positions. The difference of costs between the two algorithms increases with the number of trains departed and is on average 20% at the end of the simulation process.

Fig.4 represents the reshuffling costs for the unloading and loading operations of each arrived and departed train, respectively. It is worth noticing how the unloading operations incurred by implementing the double genetic algorithm are much costlier for the first trains arrived, when more ITUs are repositioned. Such an extra-cost reduces as more trains are handled. The difference between double and single genetic inverts at the 10th train.

By summing the costs incurred in loading and unloading operations, the total costs corresponding to the application of the double genetic algorithm are on average by the 5% lower than those resulting from the application of the single genetic one.

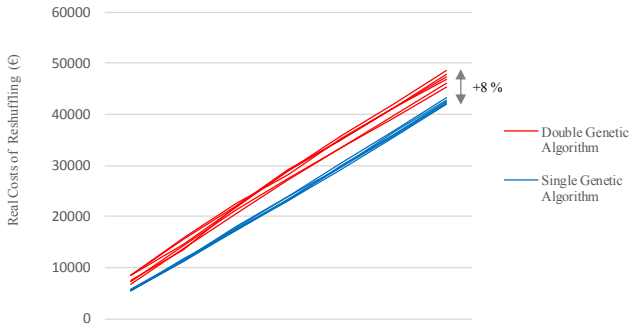

Fig. 2. Total actual costs for ITU reshuffles incurred over several simulation runs in the unloading phases corresponding to the implementation of either the single or the double genetic algorithms.

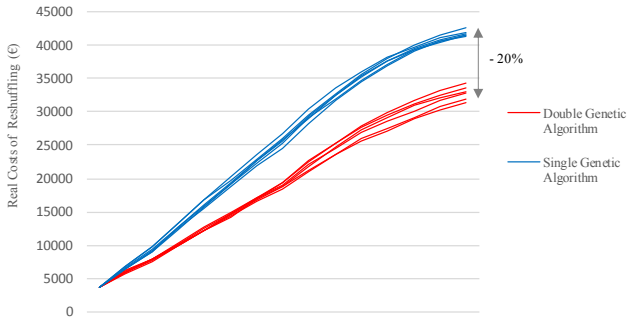

Fig. 3. Total actual costs for ITU reshuffles incurred over several simulation runs in the loading phases corresponding to the implementation of either the single or the double genetic algorithms.

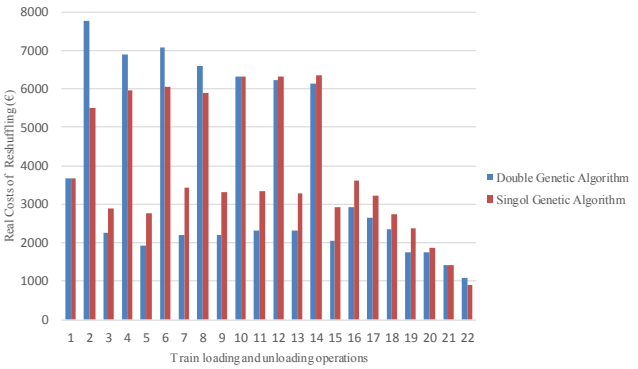

Fig. 4. Actual average costs on different simulation runs incurred in each train unloading (even train numbers up to #14) and loading operations (all remaining trains) for the single and the double genetic algorithms.

VI. Conclusions

In the paper, a solution procedure for the optimization of the reshuffles of ITUs at an inland terminal has been presented. The innovation of the procedure is the introduction of a double genetic algorithm that optimizes first the positions of the unloading ITUs and then those of the blocking ITUs that need to be reshuffled. The solution procedure uses costs that are expected to be incurred in the future when the unloaded and reshuffled ITUs will be retrieved. The single and double algorithms have been tested in simulation by computing the costs actually incurred during the entire loading and unloading process of yard bay. Simulation tests showed that the double genetic algorithm allows to reduce the total actual cost for ITUs by 5%.

The algorithm can be applied in a dynamic context. It is in fact straightforward to update the list of containers and their features with new information received in real-time, such as: arrivals of new container that have been stored in the stocking yard; a new list of containers to retrieve; update the expected container departure times.

The reshuffling algorithm is a part of a more general dynamic procedure that solves the optimization problem of storage and pick-up operations in an inland yard.

References

[1] Murty, K. G., Liu, J., Wan, Y. W., & Linn, R. (2005). A decision support system for operations in a container terminal. Decision Support Systems, 39(3), pp. 309-332.

[2] Caserta, M., Schwarze, S., & Voß, S. (2011). Container rehandling at maritime container terminals. In Handbook of terminal planning, pp. 247-269. Springer New York.

[3] Tang, L., Jiang, W., Liu, J. & Dong, Y. (2015). Research into container reshuffling and stacking problems in container terminal yards, IIE Transactions, 47:7, 751-766.

[4] Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014). Storage yard operations in container terminals–Literature overview, trends, and research directions. European Journal of Operational Research, 235(2), 412-430.

[5] Yang, J. H., & Kim, K. H. (2006). A grouped storage method for minimizing relocations in block stacking systems. Journal of Intelligent Manufacturing, 17(4), 453-463.

[6] Wan, Y. W., Liu, J., & Tsai, P. C. (2009). The assignment of storage locations to containers for a container stack. Naval Research Logistics (NRL), 56(8), 699-713.

[7] Kim, K. H., & Hong, G. P. (2006). A heuristic rule for relocating blocks. Computers & Operations Research, 33(4), 940-954.

[8] Caserta, M., Schwarze, S., Voß, S. 2009. A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. In: Cotta, C., Cowling, P., (Eds.), Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol. 5482, pp. 37–48.

[9] Lee, Y., & Lee, Y. J. (2010). A heuristic for retrieving containers from a yard. Computers & Operations Research, 37(6), 1139-1147.

[10] Kang, J., Ryu, K. R., & Kim, K. H. (2006). Deriving stacking strategies for export containers with uncertain weight information. Journal of Intelligent Manufacturing, 17(4), 399-410.

[11] Zhao, J., & Tang, L. (2011). The simultaneous quay crane and truck scheduling problem in container terminals. In 2011 International Conference on Intelligent Computation Technology and Automation (ICICTA), Vol. 1, pp. 279-282).

[12] Borgman, B., van Asperen, E., & Dekker, R. (2010). Online rules for container stacking. OR spectrum, 32(3), 687-716.

[13] van Asperen, E., Borgman, B., & Dekker, R. (2013). Evaluating impact of truck announcements on container stacking efficiency. Flexible Services and Manufacturing Journal, 25(4), 543-556.

[14] Dekker, R., Voogd, P., & van Asperen, E. (2006). Advanced methods for container stacking. OR spectrum, 28(4), 563-586.

[15] Carrese, S., & Tatarelli, L. (2011). Optimizing the stacking of the Intermodal Transport Units in an inland terminal: an heuristic procedure. Proc.Social Behav. Sci., 20, pp. 994-1003.

[16] Pap, E., Bojanic, G., Ralevic, N., Georgijevic, M., & Bojanic, V. (2012). Crane scheduling method for train reloading at inland intermodal container terminal. In IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics (SISY), pp. 189-192.