E-CREM: A Specialized Evolutionary Algorithm for the Multiple Bay Container Relocation Problem

Camila Diaz
Universidad Técnica Federico Santa María
Valparaíso, Chile
camila.diaz@usm.cl

Scarleth Bazaes
Universidad Técnica Federico Santa María
Valparaíso, Chile
scarleth.bazaes@usm.cl

Maria Cristina Riff
Universidad Técnica Federico Santa María
Valparaíso, Chile
maria-cristina.riff@usm.cl

***Abstract*—The Container Relocation Problem for multiple bays consists of finding the minimum number of moves to load a set of stacked containers on a ship according to a given loading sequence and minimizing the crane's working time for a yard of multiple bays. In this paper, we propose an Evolutionary Metaheuristic Algorithm to solve this problem. We use a heuristic encoding representation that allows us to solve the problem of solution's reconstruction that arises when applying mutation and crossover operations on traditional movements representations. In order to validate our approach, we use a large set of well-known instances, as well as a statistical analysis of our results. Our experiments show that our approach obtains high quality solutions, minimizing both goals in a reduced computational time especially for large size instances.**

I. Introduction

In the multiple bay container relocation problem there is a storage area (the yard) formed by bays and each bay has a set of stacks of containers. The retrieval process requires to take the stored containers in a given sequence. A sequence is the ordered list of containers identifications (numbers). The sequence of containers to be loaded on a given ship is known only when the ship is docked. Therefore, there are no possible pre-relocations. The goal is to minimize the number of container relocations and the total crane's working time in order to retrieve all the containers from the yard according to the sequence.

In Section II we present an overview of the state of art for the multiple bay container relocation problem and a discussion of the motivation behind our approach. A classical description of the problem as well as the associated constraints are presented in Section III. In Section IV we present our specialized approach, which is an evolutionary algorithm that uses a new representation that allows us to reduce valuable execution time and also utilize multiple heuristics in order to improve the two objectives. The experimental configuration is presented in Section V and the evaluation of our approach in Section VI, which includes a performance comparison using well-known benchmarks and a statistical comparison with the best algorithm from the literature. Finally, in Section VII we present our conclusions and future work.

II. Related Work

The first approach to the multiple bay problem was presented by Lee and Lee [1], who developed a three-phase heuristic to solve this problem. The optimization goal is to minimize the total number of movements and the crane's working time. The first phase consists of the creation of an initial feasible movement sequence using a greedy strategy. The second phase tries to reduce the number of moves of the first phase through an iterative binary integer program. Then, in the third phase, they use a mixed integer program to reduce the crane working time. Their results show that the approach was able to solve instances with more than 700 containers. However, the computational times were too high to use the heuristic in real world applications.

Later, Caserta, Schwarze & Voß [2] define more precisely the conditions of the Container Relocation Problem, in order to verify that the complexity of this problem is $\mathcal{NP}$*-hard*. This is done by establishing that the Container Relocation Problem is a special case of *Block-World Problem* (BWP) which is known to be $\mathcal{NP}$*-hard* since 1992 [3]. They also introduce two mathematical models and solve some instances using *Branch & Bound* for both models, and present a heuristic aimed at establishing an upper bound for container relocations.

Forster & Bortfeld [4] present a tree search procedure that defines well and bad placed containers for a better calculation of a lower bound, and develop a classification scheme to differentiate between productive and unproductive movements for the containers. The procedure builds sequences of movements and branches only those that have a promising outcome, limiting the width and height of the tree search. An upper bound is created by a greedy algorithm before the execution of the tree search.

Bian & Jin [5] present a three phase hybrid algorithm where the first phase develops a sequence of feasible movements that allow the crane to retrieve all containers from the yard without any problems. The containers are relocated into a new stack by different heuristics depending on the conditions of the operation of the yard. The second phase considers the sequence generated by the first phase and tries to find alternative moves according to their feasibility. The third phase uses dynamic programming to improve the sequences in phase 2 by building a network without loops for all the sequences to shorten the steps of the problem. Afterwards, it uses dynamic programming to solve the problem. Their results outperform those presented in [1], reducing both objectives and significantly improving the computational times required.

Lin, Lee & Lee [6] present a heuristic based on groups to solve the problem with cranes that can move one container at a time and cranes that can move more than one container at a time. They also presented the possibility of retrieving more than one container at once, called a group. However, all containers in a group must be retrieved in order to continue with the next group. The traditional container relocation problem works with one retrieving at a time, which means that each container belongs to only one group. To apply the heuristic based on groups, the authors incorporated a virtual container at the lower part of each stack that cannot be relocated or retrieved.

All of the previous approaches solved the Container Relocation Problem with multiple bays using heuristics, and most of them uses separated steps to deal with two different subproblems. One step minimizes the number of movements. The other step shortens the crane's working time. However, optimizations by step can be computationally time consuming.

In [7] Cifuentes & Riff proposed a metaheuristic approach that addresses both minimization objectives at the same time. The metaheuristic used was a Greedy Randomized Adaptive Search Procedure (GRASP) with two phases. First, a constructive phase using a myopic function and a restricted candidate list. And second, a local search phase based on a hill climbing algorithm to improve the solutions by exploring nearby solutions. The approach was tested in the random instances presented in [1], introducing new lower bounds for some of the instances, and reducing computational costs.

Đurasević & Đumić [8] presented a hyper-heuristic approach based on Genetic Programming (GP) to design new automatic heuristic relocation rules for the restricted and unrestricted variants of the problem. The algorithm uses a relocation schema that defines how the solution is constructed and a priority function to determine the relocation decision based on a list of available relocation strategies. Their results show that the relocation rules generated automatically outperformed the cases when using manually designed heuristics, especially when strategies for crane time optimization are introduced, indicating that the existing heuristics are not suitable to optimize this criterion. However, the GP approach needs a training and validation process that required a large set of instances, which must be done prior to using the algorithm in a final instance. This required process can take between 96 and 137 minutes for the container relocation problem with multiple bays.

More recently, Bazaes et al. [9] proposed an evolutionary algorithm to solve the Container Relocation Problem with multiple bays considering only minimization of movements. The authors also introduced a new representation that allows them to keep feasible solutions after applying any kind of operator by encoding the relocation procedure throughout the heuristic that needs to be used in every relocation. The approach was able to obtain high quality solutions with a very reduced computational cost, especially on medium to large size instances. The results presented in this research had motivated us to propose an evolutionary algorithm capable of using multiple heuristics with this new representation, and considering the movements and the crane's working time minimization.

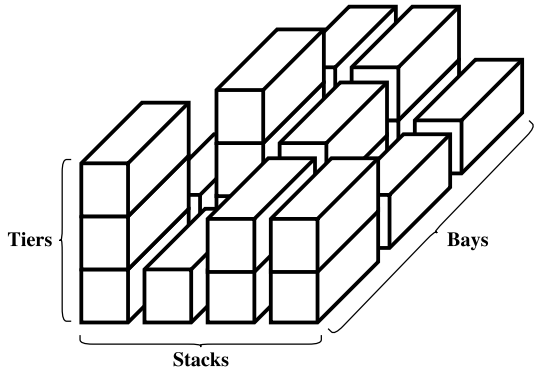

Fig. 1. Operation yard.

III. Problem Description

Our study focuses on grounded storage operations in which containers are stacked on the ground in several levels. This grounded space is called a yard and is divided into several blocks or bays. A bay consists of a set of stacks with containers piled up in tiers. Figure 1 illustrates these terms. Since containers are piled, only the top container of each stack can be accessed. If another container needs to be moved, containers that are placed on top of the required one have to be relocated.

The objective of yard optimization is to minimize retrieval times of containers while maximizing storage space utilization. To achieve this goal, several problems have to be addressed. Inbound and outbound containers have to be allocated to storage blocks while balancing the workload between blocks. Then, each container has to be assigned to a slot within the block. The objective is to avoid relocations at a later time. However, little information about the retrieval time of a container is known at the time it has to be stored, and relocations cannot be avoided. Container terminals use less busy periods to rearrange the yard as new information becomes available. This rearrangement cannot always be made. If the yard is not prearranged, the objective is to retrieve containers from the bay in the given sequence as fast as possible. The aim is to reduce the number of relocations necessary during the subsequent retrieval process. We are interested in the problem with several bays.

The Container Relocation Problem for Multiple Bays has been introduced by Lee & Lee [1]. It consists of defining a container retrieval planning associated to a pre-established list of containers, considering all the bays in the operation yard, and allowing relocations among bays. The main objective of the container relocation problem for multiple bays consists in retrieving all containers from the yard while minimizing the number of container relocations and the working time of the cranes. This problem can be defined as follows:

Given:

Algorithm 1 Main Structure of E-CREM Algorithm

```
Initialize random population
Evaluate initial population
for generation = 1 TO max_generations do
    Generate new population
    Evaluate new population
end for
Select best individual
```

Fig. 2. An example of a solution representation.

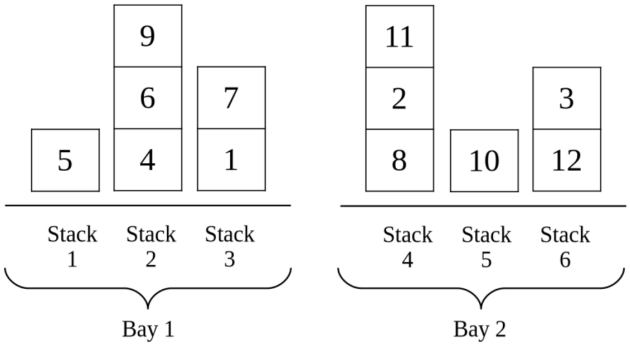

Fig. 3. Reduced yard example of 2 bays.

- A storage yard with B bays. Each bay has W stacks of containers (width).
- A total number of N containers, all of the same size

The following constraints are:

- Each stack can store a maximum of H containers (height constraint).
- Only the container from the top of the stack can be moved.
- A container can only be placed on top of a stack or on the ground within the storage bay.
- A crane can lift only one container at a time.

IV. E-CREM

Our approach is an Evolutionary algorithm to solve the container relocation problem for multiple bays (E-CREM). In the literature, the most used representation contains explicit container movements for retrieval or relocation. With this representation, each time a modification is made to the solution, a reconstruction process is needed to adjust it. In order to avoid this problem, the representation was changed to contain only the heuristics applied to the relocation movements, as presented in [9]. This adaptation allows us to implement the evolutionary algorithm in which the individuals within a population correspond to a list of the heuristics needed to extract all the containers in the yard. The pseudo-code in Algorithm 1 shows the main structure of our E-CREM implementation[1].

A. Representation

In order to represent a solution in our approach, a list of numbers correlated with a specific heuristic was used, as shown in Figure 2. This heuristics are only applied when a relocation is needed and determine to which stack the blocking container must be relocated.

To help understand how the heuristics were applied, the third dimension of the yard was reduced, placing all the bays next to each other towards the right (See Figure 3), but always considering the final retrieval point on the left of each bay.

[1]Full code implementation can be found at https://github.com/camiladc/E-CREM-CEC2026

The heuristics used in E-CREM are the following:

1) **Myopic Space**: Consists in reducing the average height of the stacks, through the relocation of containers to the stacks with more space. In this heuristic, the first stack from left to right with the highest available space is selected, excluding the origin stack.
2) **RI**: Also known as the Reshuffle Index, this heuristic computes the number of containers that should be extracted before the current container for every available stack. Similarly to the previous heuristics, the first stack from left to right with the lowest value is selected.
3) **RIL**: The Reshuffle Index with Look-ahead (RIL), is an extension of the RI heuristic, where the difference resides in the breaking ties selection. RIL uses a look-ahead mechanism, selecting the first stack in which the highest priority of retrieval is the lowest. The main goal of this rule is to avoid additional relocation in the near future.
4) **RIR**: A modification of the RI heuristic that allows to break ties in the reverse order, selecting the farthest to the right stack with the lowest value.
5) **RILR**: Similar to the previous modification, in this case for the RIL heuristic, that allows to break ties in the reverse order, selecting the farthest to the right stack where the highest priority of retrieval is the lowest.

B. Evaluation function

To ensure that both objectives of the problem are minimized: the amount of movements and the working time of the crane, our evaluation function is the same as that proposed in [7].

First, a function that counts the number of movements. As the number of retrievals will be constant for each instance and because of the representation used, we will only count the length of the solution that is equivalent to the required number of relocations as shown by the formula (1).

$$Eval\_F_1 = sizeof(solution) \tag{1}$$

Second, a function that calculates the time that the crane takes to perform all the relocations and retrievals until the yard is emptied. For this function, the following variables were

used to calculate the amount of time to make a movement i in formula (2):

- b : Number of bays displaced in the movement.
- t_{ad} : Set-up time needed by a crane to move from one bay to another. In this work $t_{ad} = 40$ seconds.
- t_{pp} : Set-up time required by a crane to retrieve a container. In this work $t_{pp} = 30$ seconds.
- t_b : Additional time to t_{ad} when moving a container from one bay to another. In this work $t_b = 3.5$ seconds.
- s : Number of stacks displaced in the movement.
- t_s : Additional time to t_{pp} when moving a container from one stack to another. In this work $t_s = 1.2$ seconds.
- ad : Boolean that shows if there was a change of bay in the movement. Its value is 1 if $b \geq 1$ and 0 in any other case.

$$time(i) = b * t_b + s * t_s + ad * t_{ad} + t_{pp} \quad (2)$$

The second part of the evaluation function is expressed by formula (3) and corresponds to the sum of the time of all the movements applied.

$$Eval\_F_2 = \sum_i time(i) \quad (3)$$

Finally, formula (4) corresponds to the complete evaluation function where α and β are weights for each objective.

$$Evaluation\_Function = \alpha * Eval\_F_1 + \beta * Eval\_F_2 \quad (4)$$

C. Initial Population

Each individual in the initial population is constructed by resolving the instance by randomly selecting relocation heuristics, where each heuristic has the same probability of being selected. This procedure allows us to create feasible solutions.

D. Selection

To select which individuals in the current population will become parents for the next generation, a tournament is applied. For each pair of individuals, their evaluation functions are compared and the individual with the best value is selected. In each generation, the best individual is passed directly to the next generation, to avoid losing a potential best solution.

E. Operators

A total of two crossover and three mutation operators were used to create the next generation of individuals.

- **One-point crossover**: A point to divide two parents is randomly selected. Then, two offspring are created by mixing the divided sections of the parents in an interleaved manner.
- **Two-point crossover**: Two points to divide two parents are randomly selected. Then, two offspring are created by mixing the divided sections of the parents in an interleaved manner.

Algorithm 2 E-CREM: Generate new population procedure

```
Copy elite individuals to new population
for i = elite_count TO population_size - 1 do
    parent1 = tournament selection
    if random() < p_cross then
        parent2 = tournament selection
        if random() < p_crossOne then
            children = one_point_crossover(parent1, parent2)
        else
            children = two_point_crossover(parent1, parent2)
        end if
        Add children to new population
    else if random() < p_cross + p_mut then
        if random() < p_mutSwap then
            Apply swap mutation
        else if random() < p_mutSwap + p_mutInversion then
            Apply inversion mutation
        else
            Apply intFlip mutation
        end if
        Added mutated parent1 to new population
    else
        Copy parent1 to new population
    end if
end for
```

- **Swap mutation**: Each heuristic belonging to the individual has a probability p_{swap} of being exchanged for another heuristic chosen randomly from the same individual.
- **Inversion mutation**: Two positions of the individual are randomly selected and the heuristics within the selected interval are reversed.
- **IntFlip mutation**: Each heuristic belonging to the individual has a probability $p_{intFlip}$ of being replaced by a randomly chosen heuristic.

In addition to the previous probabilities $p_{intFlip}$ and p_{swap}, four more were considered:

- p_{cross} : Probability of applying a crossover operator between two parents.
- $p_{crossOne}$: Probability of applying the One-point crossover operator.
- p_{mut} : Probability of applying a mutation operator.
- $p_{mutSwap}$: Probability of applying the Swap mutation operator.
- $p_{mutInversion}$: Probability of applying the IntFlip mutation operator.

For the Two-point crossover operator, the probability can be calculated as $1 - p_{crossOne}$. Similarly, for the IntFlip mutation operator, the probability will be $1 - (p_{mutSwap} + p_{mutInversion})$. During the new population procedure, a random value between 0 and 1 is always obtained to evaluate these probabilities. A more detailed explanation of how the probabilities are used, to select the operator that will be applied to each individual, is presented in Algorithm 2.

TABLE I
PARAMILS PARAMETERS CONFIGURATION.

Parameter	Possible values	Best value
α	[0.1, 0.5, 1, 5, 10]	5
β	[0.0005, 0.001, 0.005, 0.01, 0.05]	0.001
$maxGen$	[600, 800, 1000, 1500, 2000]	1500
$popSize$	[15, 20, 25, 30, 35]	30
p_{cross}	[0.1, 0.2, $\cdots$, 0.8, 0.9]	0.4
$p_{crossOne}$	[0.1, 0.2, $\cdots$, 0.8, 0.9]	0.3
p_{mut}	[0.1, 0.2, $\cdots$, 0.8, 0.9]	0.7
$p_{mutSwap}$	[0.1, 0.2, $\cdots$, 0.8, 0.9]	0.5
$p_{mutInversion}$	[0.1, 0.2, $\cdots$, 0.8, 0.9]	0.3
p_{swap}	[0.1, 0.2, $\cdots$, 0.8, 0.9]	0.4
$p_{intFlip}$	[0.1, 0.2, $\cdots$, 0.8, 0.9]	0.9

V. EXPERIMENTS

To evaluate our proposed algorithm, we performed several experiments and compared the results with those found in the literature.

E-CREM is programmed in C++ and compiled with `gcc`. The tests were run on a computer with an Intel(R) Xeon(R) CPU E5-2680 v3 processor with 64GB of RAM and a Linux operating system.

We used a sub-set of the well-known instances created by Lee & Lee [1] as training instances.

These instances are divided into two types:

- *Random* (R) : The containers are randomly located in the operation yard. (5 instances per bay configuration, 50 total instances)
- *Upside Down* (U): The containers with the highest priority to be retrieved are in the lower part of the stacks in the operation yard. (2 instances per bay configuration, 20 total instances)

We use ParamILS [10] to tune the value of the parameters α and β of the evaluation function, the seven parameters required for the operators, and also the $maxGen$ (maximum number of generations) and $popSize$ (population size) required for the evolutionary algorithm.

The best configuration found by ParamILS is presented in Table I. Based on these results and the structure of the new population procedure, all operators will be activated, and only the direct copy of the parent to the new population presented in line 23 of Algorithm 2 will never be executed.

Finally, for each instance, we have executed the algorithm ten times using different random seeds.

VI. RESULTS

We compare our results with the benchmarks of L&L [1], B&J[5] and G-CREM [7]. To our knowledge, only these authors have proposed approaches that optimize both minimization objectives and present complete numerical results, except for the upside-down instances that are only present in [1]. We also take into account the difference between processors in each work. Thus, the execution times presented in this section are normalized according to the performance of each CPU used in the approaches to which we compare our solution[2].

TABLE II
PERFORMANCE COMPARISON ON CRANE TIME PER MOVEMENT (T/M).

instance	L&L	B&J	G-CREM	E-CREM	Gap (%)
R011606	91.56	100.15	57.16	45.56	-20.29%
R011608	88.58	114.48	58.53	45.06	-23.00%
R021606	95.39	99.76	84.45	67.88	-19.63%
R021608	92.98	106.74	87.45	70.37	-19.53%
R041606	100.62	101.33	102.53	84.86	-15.67%
R041608	96.28	116.31	104.12	91.04	-5.44%
R061606	105.53	106.63	111.30	97.33	-7.78%
R061608	98.89	129.69	113.48	102.18	3.33%
R081606	111.03	114.78	117.91	103.71	-6.60%
R101606	116.96	98.35	122.70	109.41	11.24%
U011606	90.51			42.95	-52.55%
U011608	90.00			44.32	-50.76%
U021606	93.31			62.41	-33.12%
U021608	91.02			65.88	-27.62%
U041606	98.61			83.94	-14.87%
U041608	95.24			87.70	-7.92%
U061606	103.27			95.80	-7.24%
U061608	99.30			99.02	-0.29%
U081606	108.92			104.46	-4.09%
U101606	113.84			110.08	-3.30%

To simplify the comparison, all results were averaged to be presented per bay configuration and instance type, and the crane's working time per movement ratio (T/M) is introduced to compare the results more effectively.

Taking into account the T/M ratio calculated in table II, our approach is able to reduce this ratio in almost all instances. We report our gap with respect to the best known results so far.

$$Gap = \frac{\text{(E-CREM result)} - \text{(best known result)}}{\text{(best known result)}} \quad (5)$$

Also, in all the *Upside Down* (U) instances our approach is able to reduce the execution times, and for the *Random* (R) instances the average Delta is lower than 20 seconds (table III).

Additionally, we used the pair-wise Wilcoxon test to compare the performance of the T/M ratio obtained by our approach against each algorithm. For this analysis, we used the total set of instances without aggregation per bay configuration.

From table IV when we consider a significance level of 0.05 we observe significant statistical evidence that E-CREM outperforms all previous approaches as we keep the optimization goal in both minimization objectives.

VII. CONCLUSIONS

In this work, we have proposed an evolutionary algorithm that focuses on both minimizing the moves and the crane's working time. We have adapted the problem representation to allow us to apply direct operators, ensuring feasible solutions during the entire process. This representation enabled the use of five different heuristics.

[2]Ref. https://browser.geekbench.com/processor-benchmarks

TABLE III
PERFORMANCE COMPARISON ON CPU TIME EXECUTION.

instance	L&L	B&J	G-CREM	E-CREM	Delta
R011606	3573.31	1.35	0.15	2.25	2.11
R011608	5815.92	3.57	0.58	4.03	3.45
R021606	9398.76	5.39	0.77	6.66	5.89
R021608	9376.18	5.59	3.74	14.06	10.32
R041606	9373.74	91.57	8.44	21.95	13.51
R041608	9246.98	83.88	26.87	47.13	20.26
R061606	9302.61	101.52	16.32	47.86	31.55
R061608	9091.48	96.42	115.55	99.83	3.41
R081606	9233.87	100.92	31.33	79.81	48.48
R101606	9078.73	144.49	76.82	118.37	41.55
U011606	6221.63			2.73	-6218.90
U011608	6447.76			4.22	-6443.54
U021606	9383.18			8.34	-9374.84
U021608	9307.94			14.55	-9293.39
U041606	9286.58			26.96	-9259.61
U041608	9098.59			49.18	-9049.41
U061606	9115.73			56.55	-9059.18
U061608	8739.83			106.09	-8633.74
U081606	8930.38			98.17	-8832.21
U101606	8589.91			148.77	-8441.15

TABLE IV
WILCOXON TEST SET WITH RESPECT TO THE RATIO CRANE TIME PER MOVEMENT (T/M).

Comparison	Neg. Ranks	Pos. Ranks	Ties	p-value
R instances				
L&L - E-CREM	4	46	0	<.001
B&J - E-CREM	1	49	0	<.001
G-CREM - E-CREM	0	50	0	<.001
U instances				
L&L - E-CREM	0	20	0	<.001

We have tested our approach using a large set of instances from the literature for the Container Relocation Problem for Multiple Bays. Our algorithm E-CREM is able to improve most of the best-known results. Moreover, our algorithm shows robust behavior using different seed values and reduced computational times.

Our future work will focus on a more in depth analysis of the algorithm operators, the direct application of the T/M ratio as the evaluation function, and also exploring other variants of the problem and the impact of different heuristics applied under specific circumstances,

ACKNOWLEDGMENT

The authors would like to thank the support of the project Fondecyt 1241112.

REFERENCES

[1] Y. Lee and Y.-J. Lee, "A heuristic for retrieving containers from a yard," *Computers & Operations Research*, vol. 37, no. 6, pp. 1139–1147, Jun. 2010. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0305054809002433

[2] M. Caserta, S. Schwarze, and S. Voß, "A mathematical formulation and complexity considerations for the blocks relocation problem," *European Journal of Operational Research*, vol. 219, no. 1, pp. 96–104, May 2012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0377221711011337

[3] N. Gupta and D. S. Nau, "On the complexity of blocks-world planning," *Artificial Intelligence*, vol. 56, no. 2, pp. 223–254, Aug. 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000437029290028V

[4] F. Forster and A. Bortfeldt, "A tree search heuristic for the container retrieval problem," in *Operations Research Proceedings 2011*, D. Klatte, H.-J. Lüthi, and K. Schmedders, Eds. Berlin, Heidelberg: Springer, 2012, pp. 257–262.

[5] Z. Bian and Z.-h. Jin, "Optimization on Retrieving Containers based on Multi-phase Hybrid Dynamic Programming," *Procedia - Social and Behavioral Sciences*, vol. 96, pp. 844–855, Nov. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042813022222

[6] D.-Y. Lin, Y.-J. Lee, and Y. Lee, "The container retrieval problem with respect to relocation," *Transportation Research Part C: Emerging Technologies*, vol. 52, pp. 132–143, Mar. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X15000327

[7] C. D. Cifuentes and M. C. Riff, "G-CREM: A GRASP approach to solve the container relocation problem for multibays," *Applied Soft Computing*, vol. 97, p. 106721, Dec. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1568494620306591

[8] M. Đurasević and M. Đumić, "Designing relocation rules with genetic programming for the container relocation problem with multiple bays and container groups," *Applied Soft Computing*, vol. 150, p. 111104, Jan. 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1568494623011225

[9] S. Bazaes, B. Wohlwend, C. Bracamonte, C. Saavedra, N. Paz, and M. C. Riff, "Container Relocation using an Evolutionary Metaheuristic Algorithm," in *2025 44th International Conference of the Chilean Computer Science Society (SCCC)*, Oct. 2025, pp. 1–8, iSSN: 2691-0632. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11420569

[10] F. Hutter, T. Stuetzle, K. Leyton-Brown, and H. H. Hoos, "ParamILS: An Automatic Algorithm Configuration Framework," *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, Oct. 2009, arXiv:1401.3492 [cs]. [Online]. Available: http://arxiv.org/abs/1401.3492