SCIENTIFIC JOURNAL
OF MARITIME RESEARCH
[POMORSTVO]

University of Rijeka
FACULTY OF MARITIME STUDIES

https://doi.org/10.31217/p.38.2.5

Efficient Space Allocation for Import Containers Minimizing Reshuffling in Yard Blocks: Case Study

Hizia Amani[1*], Linda Bouyaya[1], Rachid Chaib[1], Mohamed Seghir Amani[2]

[1] Transport Engineering Department, LITE Laboratory, University of Constantine 01, Algeria, e-mail: hizia.amani@doc.umc.edu.dz; bouyayalinda@yahoo.fr; r3chaib@yahoo.fr

[2] Electronics and Computer Science Department, University of Houari Boumediene, Algeria, e-mail: Amanimohamed023@gmail.com

[*] Corresponding author

ARTICLE INFO

Original scientific paper
Received 2 July 2024
Accepted 26 August 2024

Key words:
Maritime transport
Container storage problem
Block allocation problem
Container reshuffling
Retrieval
Genetic algorithm

ABSTRACT

Ports play an essential role in international trade, and any inefficiency can lead to costly delays and major disruptions. Therefore, the efficiency of port operations is crucial for the smoothness of global supply chains helping reduce the risk of congestion and accidents, thereby enhancing maritime safety. Optimizing container management is particularly significant in this context. Thus, the problem of storage space allocation problem is a critical aspect of managing port operations, significantly influencing the efficiency of the retrieval process and the number of reshuffles. This study proposes a new policy, aimed at optimizing the use of space for import containers in a constrained environment. This policy incorporates considerations to facilitate the retrieval process and reduce the expected number of reshuffles. As a case study, we have taken the Port of Annaba in Algeria. Using an improved genetic algorithm and a heuristic approach, we achieved up to 33% improvement compared to the existing port policy. This demonstrates the effectiveness of our policy in enhancing port operations within spatial constraints.

1 Introduction

The growing evolution of maritime trade has elevated container transportation to a key logistical element, positioning container terminals as pivotal nodes in international transport [1, 2]. This significant increase in global trade volume requires a comprehensive redesign of the transportation network to accommodate the intensity of cargo flow [3]. Logistics is the art of efficiently managing the flow of goods and services from their point of origin to their final destination, optimizing storage, transportation, and distribution processes. Transport plays a central role in logistics, ensuring the physical movement of goods across various stages of the supply chain [4]. The movement of goods in standardized containers using various modes such as ships, trucks, trains, or barges, facilitates seamless transitions without direct handling of the freight during mode changes [5, 6].

To ensure smooth operations at the port, including berthing, unloading/loading operations, and storage, it is imperative to effectively manage the complexity of these processes [7]. Efficient berthing schedules, accurate unloading/loading procedures, and strategic yard management are essential to minimize delays, optimize resources, and prevent congestion [8, 9].

At container terminal, the storage yard is a dynamic space where containers pause before continuing their journey by road, rail or sea. Inside the yard, containers are carefully stacked to maximize the use of available space. The precise details of container stacking reveal an important aspect of operations: as the stacks rise to the top, the equipment used to handle them generally interacts only with the highest container [10]. The storage yard consists of several perpendicular or parallel ways to the berth known as blocks. Each block is composed of a number of bays, and within each bay are stacks, which are characterized by their stack height, known as tiers. Additionally, within each stack, there are designated slots for the placement of containers (**Figure 1**) [11].

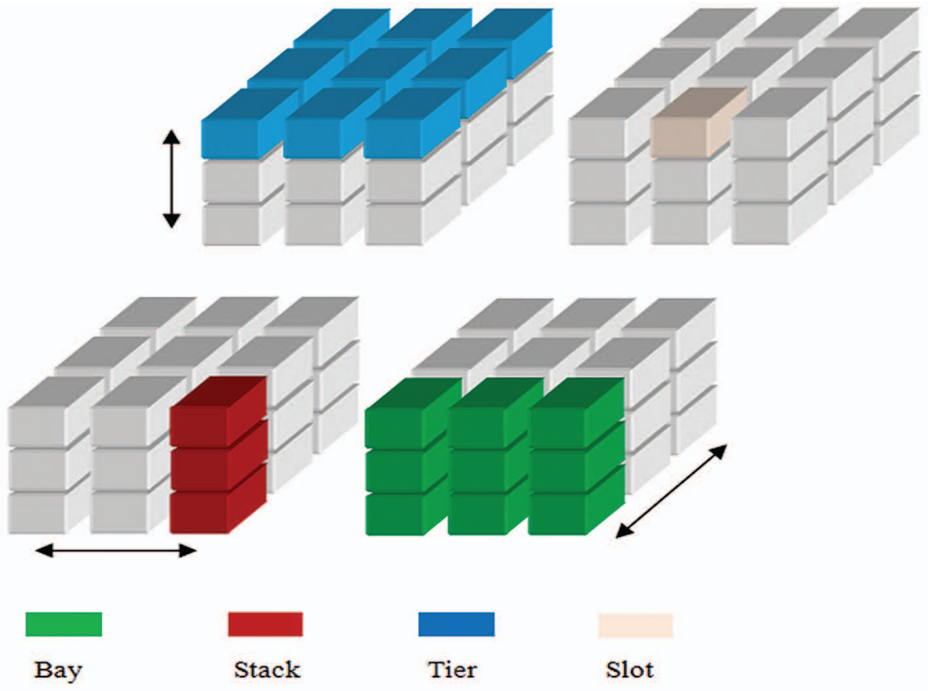

Figure 1 Storage yard layout

Small ports are facing a glaring shortage of storage space due to a significant increase in containerized traffic (as observed in the case of the port of Annaba). This saturation compromises their operational efficiency, as port facilities struggle to meet the growing demand for storage capacity. Despite exploring alternatives such as constructing storage areas away from the port, judicious optimization of existing storage space remains a crucial key, which is our objective in this work. We propose a new policy aimed at fully exploiting the available space while respecting constraints related to container retrieval and implicitly contributing to control the estimated number of reshuffles.

2 Literature review

The storage space allocation problem (SSAP) is widely discussed and often intertwined with other aspects of port operations. Therefore, addressing this issue requires consideration of these related aspects. The central objective of this problem is to optimize storage space utilization efficiently while adhering to certain rules to minimize the number of necessary reshuffles. Reshuffle, involving non-productive movements during rearrangement of containers to achieve the correct order, poses a significant challenge in port operations management. The (SSAP) has been studied extensively in the field of operations research, and various mathematical models, algorithms and heuristics have been proposed to solve it. We mention 21 studies that address the storage problem by clearly determining the context and the methods used.

Targeting export containers, where arrivals are random and containers are initially stored without specific order, it is imperative to reshuffle them based on predefined criteria such as departure date or port of destination. In this context, the studies in the literature are published to discuss the pre-marshalling problem knowing that in this problem we start with an initial layout to final layout. [12] Develop a model that adheres to a specified yard layout and follows a given sequence for loading containers onto the ship. [13] They introduce a heuristic tree search approach, the study aims to efficiently sort items into stacks based on group indices to streamline loading onto ships. Other heuristics are also developed by authors; [14] propose a heuristic solution method emphasizing the importance of prioritizing containers to minimize movements and optimize efficiency. Additionally, an instance generator is developed to create problems of varying difficulty levels. By considering factors like bay occupancy and priority container placement, instances ranging from low to high difficulty are generated. [15] Introduce two types of container, labeled as Type-A and Type-B, each requiring a sequence of container movements to meet specified conditions. Two labeling algorithms, Heuristic-A and Heuristic-B, are presented to address these problems.

This problem not only relied on using heuristics but also employed meta-heuristics. We highlight an interesting work by [16] propose a variable chromosome length genetic algorithm to solve the problem, aiming to optimize vessel loading time by minimizing mis-overlays with the fewest container movements.

To enhance the efficiency of solving instances optimally, [17] employ A* and IDA* algorithms along with various innovative branching and symmetry breaking heuristics.

While previous studies mentioned focused on minimizing crane movement, a study by [18] demonstrated that the number of moves is not a suitable indicator for measuring crane time; so authors focus on minimizing crane times rather than just the number of movements. Two exact approaches are developed: an integer linear model and a branch and bound algorithm. These methods incorporate new upper and lower bounds, dominance criteria, and a heuristic procedure to provide optimal solutions for practical-sized problems.

Similarly, when retrieving containers from the bottom of a stack, where containers above must be moved first to access the desired one, reshuffle is necessary. [19] Propose a MIP formulations and a simulated annealing algorithm to minimize the number of reshuffles in the retrieval stage. [20] Utilize a decision rule and compares it with a branch and bound algorithm. [21] Develop strategies, use a branch and bound algorithm and propose heuristics to minimize the total time of containers during the retrieval process. [22] Address the oversight of neglecting the impact of container reshuffling, incorporating the macro-level impact of reshuffling via discrete event simulation into a mixed integer programming model. Empirical methods are devised to strike a balance between computational speed and solution quality. Results highlight that disregarding reshuffling activities during planning can result in an overestimation of yard capacity.

As mentioned earlier, the storage problem has been addressed along with other major issues in container terminal operations, including yard crane scheduling; [23] propose a model to optimize both crane travel time and future relocations. Present a heuristic local search scheme based on the formulation's structure and the linear programming relaxation of subproblems. [24] Address the increasing significance of storage yard management in container terminals, with the bottleneck of port operations transitioning to the yard area. Introduce a flexible yard template strategy.

Moreover, researchers also tackled this issue using stochastic techniques, dynamic programming, and providing decision support. [25] Address space allocation for stacking export containers and propose a hybrid storage policy. They employ a stochastic programming model with the concept of recourse to construct their proposal. By combining class-dedicated and sharing strategies, the aim is to increase terminal productivity and reduce vessel duration-of-stay. [26] Propose a decision-making method to optimize container allocation within the terminal. Their model considers container arrivals and departures, as well as a certain degree of uncertainty in the retrieval order. [27] Propose a decision-based heuristic to address the storage space allocation problem in ports, aiming to minimize gantry movements while considering temporal asymmetries in container handling processes.

In the case of import containers, where customers retrieve containers randomly, additional movements are inevitable during retrieval as the order of customer requests cannot be predicted in advance. [28] Compare two strategies (segregating and non-segregating) to see the expected number of reshuffle. They did not attempt to identify an optimal strategy. [29] Adopt the segregating strategy; containers unloaded during different periods are not allowed to be stored in the same bay. Aim to minimize the number of rehandles. The constant, cyclic, and dynamic arrival rate of import containers is considered suggesting a methodology, based on the Lagrangian relaxation technique, for finding the optimal solution. [30] Examine the two previously discussed strategies. The aim is to build on these earlier findings and devise specific strategies for intermediate scenarios-situations that cannot be fully addressed by either a non-segregating or segregating strategy alone. Three distinct strategies are proposed. Each strategy involves two phases: in the initial phase, containers from various ships are separated, and in the subsequent phase, each strategy employs a unique method to blend containers from different ships. Determine for each strategy the appropriate terminal conditions. This made it easier for operators to choose the convenient strategy. [31] To minimize the number of expected reshuffle, they model the optimization problem as a generalized assignment problem, and compare the performance of two methods, the integer linear programming method and metaheuristics (genetic algorithm). [32] Propose a storage space sharing strategy between a container terminal and dry port to address space shortage for import containers. Employ a multiple-objective mixed integer programming model to minimize travel distance, minimizing imbalance in number of containers, maximizing shared storage space of the dry, and propose a non-dominated sorting genetic algorithm.

While the literature review of import containers mainly focused on minimizing the estimated number of reshuffles, this work primarily focuses on minimizing space utilization during container allocation due to the limited available space. We propose a new policy aimed at adjusting port operations to reflect real-world conditions, respecting certain constraints of the retrieval process and aiming to control the estimated number of reshuffles, thus optimizing yard efficiency. To address these challenges, an improved genetic algorithm and a proposed heuristic are utilized to solve the problem effectively. These contributions highlight the practical relevance of the study within the realm of related works.

In the next section, we present the port of Annaba and the used policy for the storage of containers. Then,

we describe the problem and the proposed policy. In section four, we explain the methods used to solve the problem. Results and discussions are provided in section five. Section six summarizes the conclusion and the recommendations.

3 Problem Statement

3.1 Case study: Port of Annaba

The port of Annaba is one of the key ports in Algeria, playing a vital role in the country's international trade. Located on the eastern Mediterranean coast, it handles a significant volume of cargo, including containerized goods. The port faces substantial challenges due to space constraints, which impact its efficiency in container management. These challenges make it an ideal case study for testing new policies aimed at optimizing space utilization.

The port of Annaba ensures the storage of import and export containers of different sizes (20', 40', and 45'). The port primarily serves as an import destination for containers, while the number of containers exported is very limited. As for empty containers returned by customers, their location is SIL (Intermodal Society of Logistics), 3 kilometers away from the port. It houses the main maritime company and serves as the primary site for receiving export containers before they are transported to the port. Regarding empty containers, coordination between the SIL and the port is based on ship availability for transporting these containers and according to the stowage plan. Communication between the port expert and SIL is conducted to determine the number of empty containers that can be transported. **Table 1** shows the number of containers returned by customers during the month of January (2024): 21 ships belonging to CMA CGM Company, 3 ships to COSCO Company and 2 ships to HMM company. Based

Table 1 The return of empty containers in January (SIL Annaba)

N°	SHIP	Empty Container		Total TEU
		20'	40'	
1	ALLEGRO 20/11/2023	2	0	2
2	ATLANTIC GENEVA 16/01/2024	36	163	362
3	CONESTE 22/06/2023	0	2	4
4	CONTSHIP ACE 19/12/2023	2	168	**338**
5	CONTSHIP ACE 21/11/2023	3	0	3
6	JAGUAR 20/12/2023	28	194	**416**
7	KESTREL 27/12/2023	1	2	**5**
8	MARINA L 05/01/2024	5	31	67
9	MARINA L 08/12/2023	5	21	**47**
10	MARINA L 11/11/2023	0	1	2
11	SKYVIEW 02/11/2023	1	1	3
12	SKYVIEW 31/12/2023	2	9	**20**
13	SPICA J 02/11/2023	0	1	2
14	SPICA J 10/01/2024	30	123	276
15	SPICA J 12/12/2023	10	39	**88**
16	SPICA J 21/10/2023	0	1	2
17	SPICA J 23/01/2024	11	10	31
18	SPICA J 26/12/2023	66	186	**438**
19	SPICA J 27/11/2023	4	1	6
20	YIGITCAN A 13/12/2023	6	3	**12**
21	YIGITCAN A 14/11/2023	2	1	4
1	YAKOOT 22/01/2024	3	6	15
2	YAKOOT 25/12/2023	8	10	**28**
3	YAKOOT 29/11/2023	1	2	5
1	YAKOOT 25/12/2023	7	5	**17**
2	YAKOOT 22/01/2024	6	1	8
Total TEU		238	979	2196

on the number of returned containers, we can infer that the number of containers unloaded during December reached 1404 TEU (Twenty-foot Equivalent Unit), providing us with an overview of the monthly unloading volume. The information obtained from SIL indicates that the number of TEU remains relatively stable across months, although variations are observed between seasons.

3.2 Port policy

At the Port of Annaba; the container storage process follows a series of well-defined stages. Initially, containers are unloaded from the ship using cranes, and then deposited in the pre-disembarkation area. Subsequently, depending on the availability of transport trucks, these containers are then transported to the designated storage area.

The storage area is divided into two main sections: import and export. Within each section, containers of different sizes are stored separately (20', 40', and 45'). For each size of container, two types are distinguished: Type A and Type B. Type A containers refer to individual containers where the estimated time of their retrieval is different. Type B containers refer to container by group; where containers of the same group have the same estimated time of retrieval. The rules, according to which they store containers at the port, whatever their size, are as follows:

1. Type A and type B containers are stored in separate bays.
2. Type B containers belonging to different groups cannot be stored in the same stack.
3. Type B containers belonging to the same group must be stored in successive stacks.

Figure 2 illustrates the policy by applying the aforementioned rules with the following data:

Type A containers = 22 containers.

Type B containers = 99 containers.

Number of groups = 6;

Number of containers in each group: [10/14/18/19/20/18] respectively.

Number of stacks = 6.

Number of tiers = 3.

Bay capacity = 3∗6 = 18 slot. (For type B containers)

Bay capacity = (3∗6) – (Number of Tiers – 1) = (18 – (3 – 1)) = 16 slot (For type A containers); the capacity of the bay is restricted to respect the retrieval process. For more details see appendix A.

As illustrated in **Figure 2**, group (1) occupies 4 stacks, leaving 2 empty slots, group (2) occupies 5 stacks, with

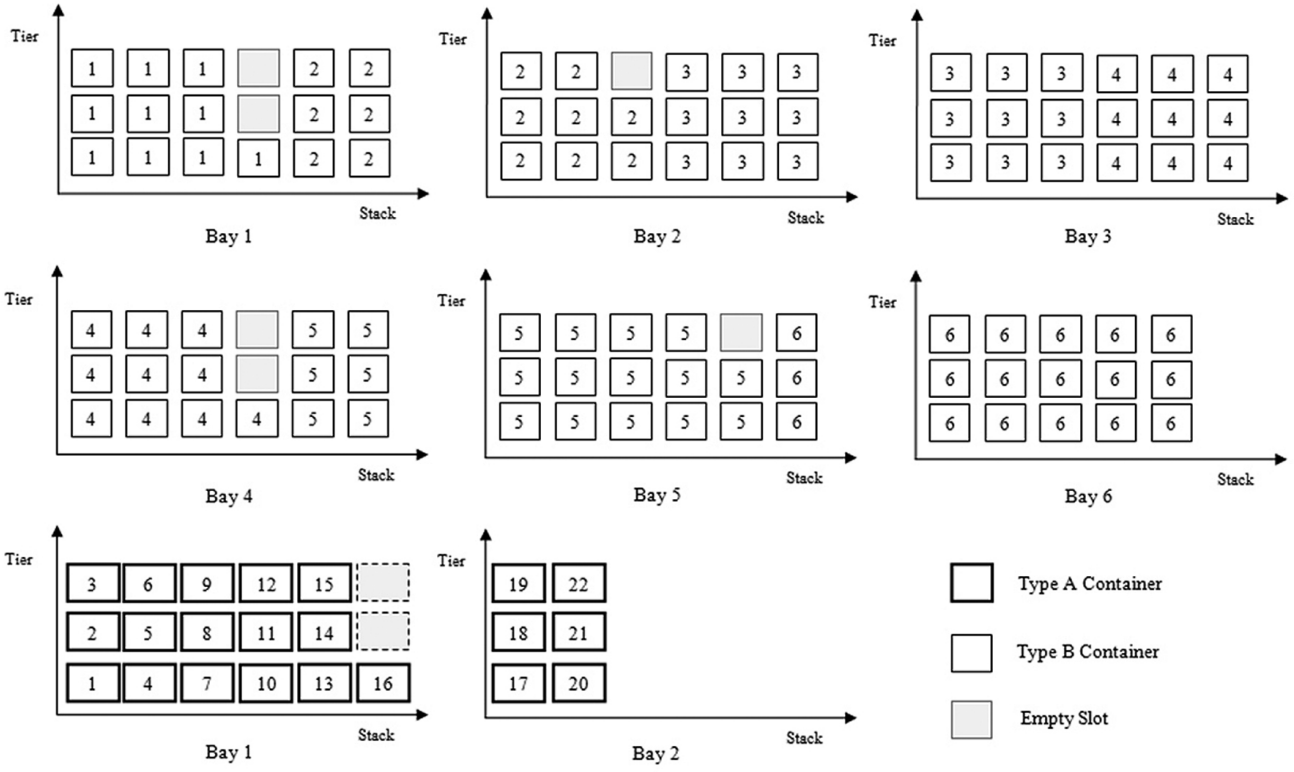

Figure 2 Storage of container according to the policy of the port

Table 2 The utilized space according to the policy of the port

Type of container	Number of utilized bays	Number of the rest stacks in the last bay	Number of empty slots
Type A	2	4	2
Type B	6	1	6
Total	8	5	8

1 empty slot remaining, and group (3) occupies 6 stacks, with no empty slot left. Group (4) occupies 7 stacks, leaving 2 empty slots. Group (5) occupies 7 stacks, with 1 empty slot remaining. Lastly, group (6) occupies 6 stacks, with no empty slots left. Consequently, the six groups collectively occupy 35 stacks with 6 empty slots. Type A containers occupies 8 stacks with 2 empty slots in the first bay. **Table 2** summarizes the results of the two types of containers.

The storage space allocation problem aims to find an optimal distribution of containers to efficiently utilize available space and facilitate container retrieval, which becomes increasingly challenging at the port of Annaba due to its restricted dimensions. Therefore, a thorough examination of the port's container storage policy is indispensable. Although the port has implemented a policy for managing container storage, it is not without flaws.

Detailing the container storage policy employed at the Port of Annaba reveals several advantages and disadvantages. Among the advantages, for type B containers, retrieval is made easier without the need for reshuffling as each group is stored in successive stacks (rule 3) and did not stored with another group in a same stack (rule 2). Additionally, the separation of type A and B containers prevents confusion between them, contributing to more efficient storage management. However, this policy also presents significant drawbacks. Firstly, it consumes more space as separating container types may require additional storage area. Moreover, type A containers often require frequent reshuffling during retrieval process, as the order of retrieval for each container is not the same. This can potentially slowing down operations and lead to delays in cargo processing. By assessing these advantages and disadvantages, it becomes crucial to strike a balance between operational efficiency and optimal space utilization to enhance the overall performance of the Port of Annaba.

3.3 Proposed policy

The fundamental principle of this policy is to store both types A and B together. The primary objective is to optimize the utilization of available space. This policy is based on two main assumptions:

1. The combined storage of the two types of containers, A and B, can benefit the use of available space.
2. The separation of type A containers into different stacks can reduce the number of reshuffles.

However, the major challenge lies in designing an efficient storage method that adheres to the container retrieval process and controls the estimated number of required reshufflings. To achieve this, we must consider the rules inherent in the port's policy:

2. Type B containers belonging to different groups cannot be stored in the same stack.
3. Type B containers belonging to the same group must be stored in successive stacks.

The first rule is relaxed to optimize the utilization of the space.

And we add:

4. Type A containers are always stored on the top of the stack.
5. If a bay contains only one group of type B containers and at least one type A container, there must be at least one container from this group at the highest tier of this bay.
6. If a bay contains more than one group of type B containers and at least one type A container, there must be an empty slot at the highest tier of this bay.

(4) The aim of reserving the highest tier for type A containers is to minimize the estimated number of reshuffles caused by storing these containers in the same bay. Storing type A containers on top of each stack facilitates their retrieval.

(5 and 6) Respect the retrieval process, specifically for type B containers, to avoid being blocked by type A containers.

Explaining the two policies clarifies that by applying the port's policy to store type A and B containers, only one way can be considered (Figure 2). However, the proposed policy requires more steps. According to constraints 5 and 6, there are several ways to store the two types of containers, especially type A containers as it is unclear in which bay and with which group these containers should be stored to achieve the final optimized location. For these reasons, we aim to find the optimal assignment of containers while respecting the mentioned rules and constraints.

4 Resolution method

The storage space allocation problem is addressed by several methods, ranging from exact methods to Meta-heuristics. For large-scale problem instances, meta-algorithms-based approaches are often favored due to their ability to provide high-quality solutions within reasonable timeframes. Among these approaches, genetic algorithms stand out as particularly affective solutions, capable of finding acceptable solution in complex and dynamic environments. This is why we have chosen the genetic algorithm to address our problem, confident in its ability to tackle the specific challenges of container management in our context. We use a local search in the crossover operator to develop the neighbors of children, aiming to explore more of the solution space and accelerate the process of finding the optimal solution. Additionally, we propose a heuristic to achieve an admissible solution. Both the algorithms and the heuristic, along with the used parameters, are presented in this section.

The algorithm is developed in PYTHON version 2.7. PYTHON is an interpreted, object-oriented, high-level programming language with dynamic semantics.

4.1 Genetic algorithm

4.1.1 Overview

Over the past forty years, biologically inspired computing has gone through various stages. Interest in this field has led to advancements in neural networks, machine learning, and evolutionary computation, particularly genetic algorithms [33]. According to [34] single-solution metaheuristics algorithms enhance a single solution through local search but may get trapped in local optima (simulated annealing, tabu search, and guided local search). Conversely, population-based metaheuristics employ multiple candidate solutions to maintain diversity and evade local optima (genetic algorithm, particle swarm optimization, and ant colony optimization).

4.1.2 Definition

A genetic algorithm is a search heuristic inspired by the process of natural selection. It falls under the broader category of evolutionary algorithms and is designed to find approximate solutions to optimization and search problems [35]. Introduced by John Holland in the 1960s, genetic algorithms draw inspiration from biological evolution, employing concepts such as selection, crossover, and mutation to evolve a population of potential solutions [36]. Over successive generations, the algorithm refines and adapts these solutions, favoring individuals that demonstrate better fitness for the given problem [37].

4.1.3 Applications

Genetic algorithms are widely applied in optimization, machine learning, and artificial intelligence. They have proven effective in solving complex problems where traditional algorithms might struggle, offering a unique approach to finding solutions through simulated evolution [38].

In the realm of supply chain management (SCM), genetic algorithms have emerged as a pivotal toot for addressing various challenges [39]. [40] Undertake a comprehensive review, encompassing 220 articles that leverage Gas across different SCM facets.

4.1.4 Genetic algorithm chromosome

A chromosome in genetic algorithms is a string of genes representing a potential solution to the optimization problem. The quality of chromosome representation in a genetic algorithm is pivotal in determining the obtained solution. A well-designed representation must balance problem complexity with algorithm efficiency, allowing genetic operators to manipulate solutions meaningfully. Inadequate choices may lead to crucial information loss or hinder convergence towards optimal solutions [41-43].

As a configuration or capacity of a bay is defined by a number of slots, and each container can be stored once (occupying one slot), the simple chromosome representation involves a container number and the affected slot number, where a slot number is denoted by bay, stack, and tier. However, opting for this representation would complicate the problem due to constraints imposed by rule 3 (successive stacks), and the solution space would be extensive. As previously mentioned, our policy's fundamental principle is to store both types A and B together. We believe a representation that integrates both container types will be more practical. Determining the optimal number of type A containers to store with groups of type B containers will be crucial, meaning the assignment of type A containers to groups of type B containers.

According to **Figure 3**, the chromosome represents the assignment of 7 containers of type A to 4 groups of containers of type B. For example, container number 2 from type A is assigned to group number 4. **Figure 4** illustrates the simplification of the chromosome representation; 2 containers of type A are assigned to group number 1. To understand the proposed policy and the application of the genetic algorithm, **Figure 5** demonstrates the storage of containers by applying the assignment represented in the representation of the chromosome.

Type A Container	1	2	3	4	5	6	7
Group Number	1	4	2	1	4	3	4

Figure 3 Representation of the chromosome (a)

Type B Container (Group Number)	1	2	3	4
Number of type A container affected to each group	2	1	1	3

Figure 4 Representation of the chromosome (b)

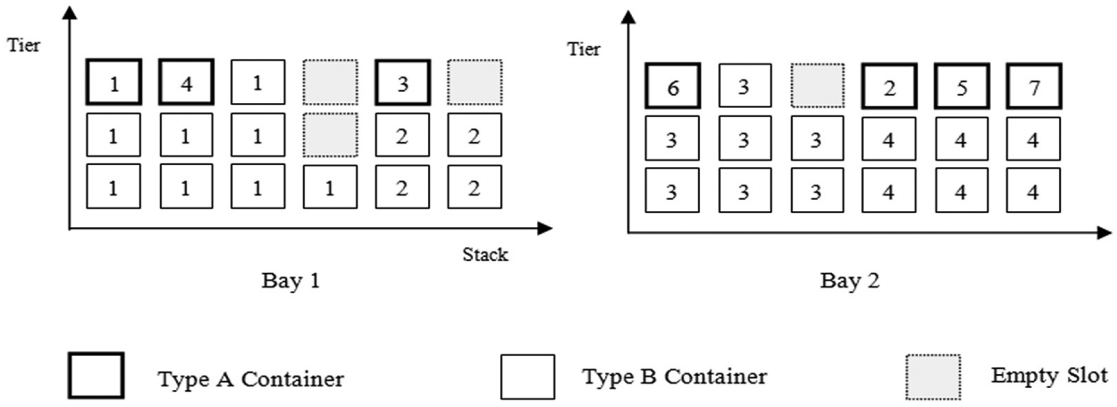

Figure 5 Storage of containers by applying the proposed policy

4.1.5 Steps

Initialization: create an initial population of potential solutions, often randomly generated [44]. It is the case of this work as we randomly generate the initial population.

Selection: evaluate the fitness of individuals in the population and choose them based on their performance [45]. The selection step in a genetic algorithm is critical for choosing the fittest individuals for reproduction. This step includes several methods, each offering unique approaches to individual selection. Among these methods are tournament selection [46], roulette wheel selection [47], rank-based selection [48] and elitism selection [49], each with its advantages and specific applications in the artificial evolution process.

We use the Rolette wheel selection; in nature, individuals best suited to their environment have a competitive edge in obtaining food and mating opportunities, thereby increasing the likelihood of passing on their genes to the next generation of the species [50].

Crossover: combine genetic information from selected individuals to create new solutions, encouraging the exchange of favorable traits [51].

We use a mono crossover and 80% for the crossover rate, this means 80% of selected parents will be operated to extract child, else we will not make crossover.

Mutation: introduce random changes to some solutions, fostering diversity in the population and preventing stagnation [52].

We use a swap mutation and 1% for the mutation rate.

Dynamic Parameter Adaptation

To avoid stagnation and encourage diversity within the population, we implemented a mechanism to adaptively adjust the mutation and crossover rates. This mechanism is triggered when all individuals in the population become identical (min=max). In such a case, the mutation and crossover rates are automatically adjusted according to the following rules:

The mutation rate is increased by 1%

The crossover rate is changed by 3%.

4.1.6 Fitness function

To evaluate the performance of the individuals, we focus on minimizing the space utilized during the allocation of containers. We identify three important components to measure this space: bays, stacks, and slots. Each component has a coefficient to express its value, with minimizing the number of bays being the most significant. The best individuals will be selected based on the minimum value.

B: Number of Utilized Bays.

S: Number of the Rest Stacks at last bay.

ES: Number of Empty Slots.

$$\text{Min } F = [100 * (B) - 10 * (S) + ES]$$

4.2 Local search

4.2.1 Definition

Local search is an optimization technique that iteratively explores neighboring solutions to improve upon the current solution. It focuses on refining solutions within a restricted, local space rather than exploring the entire solution space. The concept dates back to the mid-20th century, gaining prominence in the field of mathematical optimization [53].

4.2.2 Application

Local search algorithms have been tailored to address optimization challenges in job shop problem [54, 55, 56] and flow shop scheduling problem [57], aiming to iteratively refine schedules and minimize completion times. [58] Discuss a multi-objective genetic local search for flowshop problem, utilizing a modified local search that is applied not only after mutation but also to elite solutions from previous population. A key innovation is the limit of the number of examined neighborhood solutions, allowing for adjustable computation time. Local search techniques are also utilized for the bin packing problem [59-62].

We apply a local search after the crossover step, where the newly generated children are refined by exploring their immediate neighbors. This technique helps to improve the candidate solutions before adding them to the population.

4.3 The proposed heuristic

The problem relates to the number of groups and the number of type A containers, where the number of possibilities that can be processed equals to NG^{TA}, where NG is the number of groups and TA is the number of type A containers. However, we propose exploring the solution space effectively by proposing a heuristic aimed at finding an acceptable solution. Our heuristic method offers an effective and pragmatic approach. This approach relies on a clever distribution of type A containers, based on the capacity of each group of type B containers. Firstly, we assess the capacity of each group of type B containers, represented by the number of containers it can hold. Then, we distribute the type A containers in proportion to the capacity of each group.

Example

Type A Container = 11. Represents the (100%)

Type B Container = 50. Represents the (100%)

Number of groups and number of containers in each group = 4; [12/8/16/14]

The number of containers type A, affected to each group is:

Group 1: 11∗12/50 = 2.6 ≈ 3

Group 2: 11∗8/50 = 1.7 ≈ 2

Group 3: 11∗16/50 = 3.5 ≈ 3

Group 4: 11∗14/50 = 3.08 ≈ 3

Type B Container (Group Number)	1	2	3	4
Number of type A container affected to each group	3	2	3	3

Figure 6 Representation of the chromosome by applying the proposed heuristic

5 Results and Discussions

The results of the genetic algorithm, the heuristic and the policy of the port are presented in **Table 3**. For all instances the used capacity of bay is 18 slots; 3 tiers and 6 stacks. The total number of containers generated is between 24 and 111. The number of groups varies between 2, 3 and 4.

The GAP (1) and (2) are shown in **Table 4**.

GAP (1): between the Port's policy results and the heuristic results

GAP (2): between the Port's policy results and the genetic algorithm results.

Table 3 Results

Instances	Type A	Type B	Number of groups	Heuristic	Port's Policy	Genetic algorithm
1	4	20	2	173	223	173
2	5	24	2	181	234	181
3	6	28	2	202	242	202
4	7	32	2	263	263	263
5	8	36	2	271	324	271
6	9	40	2	292	332	292
7	10	25	3	254	267	254
8	11	30	3	263	284	263
9	12	35	3	294	334	281
10	13	40	3	354	354	354
11	14	45	3	374	374	374
12	15	50	3	463	476	463
13	10	50	4	373	396	373
14	14	56	4	455	456	402
15	18	60	4	473	526	473
16	23	65	4	555	568	502
17	29	70	4	596	649	585
18	34	77	4	676	709	649

Table 4 GAP

Instances	Heuristic	Port's policy	Genetic algorithm	GAP 1 (%)	GAP 2 (%)
1	173	223	173	22.4	22.4
2	181	234	181	22.6	22.6
3	202	242	202	16.5	16.5
4	263	263	263	0	0
5	271	324	271	16.3	16.3
6	292	332	292	12	12
7	254	267	254	4.8	4.8
8	263	284	263	7.3	7.3
9	294	334	281	11.9	15.8
10	354	354	354	0	0
11	374	374	374	0	0
12	463	476	463	2.7	2.7
13	373	396	373	5	5
14	455	456	402	<1	11.8
15	473	526	473	10	10
16	555	568	502	2	11.6
17	596	649	585	8	9.8
18	676	709	649	2	8.4

The results of the genetic algorithm and the policy of the port are presented in **Table 5**, where the number of stacks and tiers are changed. This parameter directly affects storage capacity. By adjusting these dimensions, we can better understand how variations in the physical storage configuration influence the performance of the proposed algorithm.

According to **Table 3**, the use of the genetic algorithm and heuristic has improved the port policy results in terms of minimizing the storage space used, achieving up to a 22% improvement (instance 2) (**Table 4**).

However, in some instances (particularly instances 4 and 5), we observe an equality in the results obtained. In the case where the number of containers in a group is 18 and the number of levels is 3, it means that 6 stacks are optimally used, so there will be no empty slots because 18/3 equals 6, which explains this equality and

Table 5 Impact of bay dimensions on algorithm performance

Instances	Type A	Type B	Number of stacks	Number of tiers	Port's policy	Genetic Algorithm	GAP 3 (%)
1	5	15	4	3	264	191	27.65
2	6	29	4	3	361	301	16.62
3	7	29	4	3	373	300	19.57
4	8	35	4	3	394	381	3.29
5	10	29	4	3	396	383	3.28
6	5	15	5	4	246	164	33.33
7	6	29	5	4	255	191	25.09
8	7	29	5	4	254	204	19.68
9	8	35	5	4	261	261	0
10	10	39	5	4	347	283	18.44

determines the influence of bay capacity and container grouping.

When the number of groups increases to 4 (instances 14, 16, 17, and 18), the heuristic could not achieve the same results as the genetic algorithm, indicating that its underlying principle does not always guarantee optimal results.

It is important to question the genetic algorithm's inability to improve results in equal instances. The common factor we noticed in these instances is the number of empty slots, which varies between 1 to 4. This number is too low to record significant improvements. The empty slots are not only the result of container allocation but also those left empty due to constraint 6 of our policy.

To better illustrate this hypothesis, the graph in **Figure 7** shows the impact of empty slots on improving results. We particularly observe differences in results when empty slots exceed 4. The instances used are presented in **Table 6**.

According to **Table 5**, the results show that the genetic algorithm consistently produces better outcomes, with differences ranging from 13 to 82, corresponding to an improvement of 3% to 33% in instances 4 and 6. Equality is also observed (instance 9), as in the previous results (**Table 3**).

Table 6 The utilized instances

Instances	Number of empty	Heuristic	Genetic algorithm
2	1	181	181
3	2	202	202
4	3	263	263
7	4	254	254
16	5	555	502
17	6	596	585

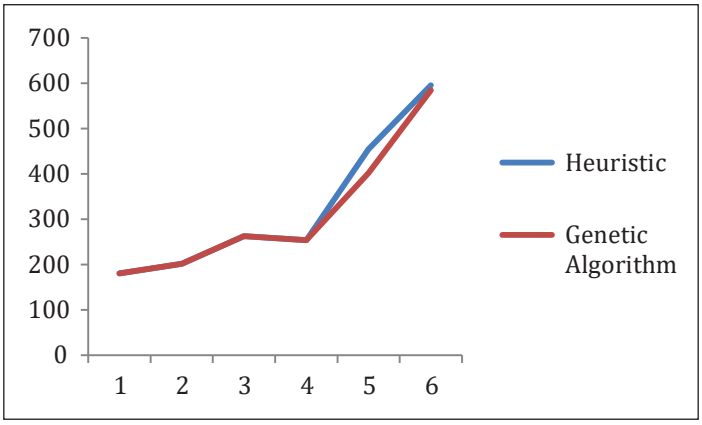

Figure 7 Impact of empty slots

The genetic algorithm's results were tested under the influence of several factors, including variations in the number of containers of each type relative to the total number, changes in the number of groups for the second type, and different bay configurations. All these factors led to changes in the improvement percentage while consistently maintaining positive improvements compared to the port's policy.

6 Conclusion

This study addresses the issue of container storage in maritime terminal, emphasizing the importance of minimizing reshuffling and ensuring the rapid retrieval of containers. The use of the genetic algorithm and heuristic has allowed us to achieve effective and significant improvements in results. This improvement is particularly important for ports with limited space, as it helps avoid delays during the container retrieval process and reduces the risk of accidents related to congestion.

Our results highlight the relationship between the number of tiers in a bay and the number of containers in each group as a determining factor. In configurations where the number of containers is perfectly divisible by the number of tiers, there are fewer opportunities for improvement.

Other ports also aim to optimize space utilization, which is a common goal. However, in these pots, the issue of reshuffling might become even more critical than space constraints. This study discusses two types of containers, A and B, representing individual and grouped containers, respectively. This classification is applicable to various ports, allowing them to benefit from this research by adapting the policy to their specific conditions. The policy can be adjusted based on the proportion of A and B containers relative to the total number of containers. Ports with a lower proportion of type A containers will benefit more from reduced reshuffling when these containers are stored above type B containers.

The problem of container reshuffling is implicitly integrated into our work; a direct extension of this problem could be addressed. We recommend shared storage between the intermodal logistics company (SIL) and the Port of Annaba, especially for containers with a high dwell time or that may exceed the free time limit.

Funding: The research presented in the manuscript did not receive any external funding.

Authors Contributions: Conceptualization: H.A.; Methodology: H.A.; Data collection: H.A.; Research and writing, H.A.; Software: M.A.; Algorithm design: M.A.; Supervision: L.B.; R.C.; Review: L.B., R.C.

Appendix A: Explanation of Retrieval and Reshuffling Process

Definition of Reshuffling

Reshuffling refers to the process of moving containers from one location to another to facilitate the retrieval of specific containers.

Capacity of bay

The capacity of a bay is defined by the formula:

$$\text{Capacity of Bay} = \text{Number of Tiers} * \text{Number of Stacks}.$$

This capacity represents the number of slots available in the bay, with each slot capable of holding a single container. During the retrieval process, to avoid confusion between containers, reshuffling is performed within the same bay. This means containers are moved from one stack to another within the same bay.

Empty Slots

To facilitate the retrieval of containers, it is essential to leave empty slots during the placement of containers. This is because the retrieval order of the containers is not known in advance, and having empty slots allows for easier reshuffling. The number of empty slots required depends on the number of tiers in the bay.

For instance, if the number of tiers is 3, then it is necessary to leave two empty slots. This is to anticipate the need to move containers that are in tier number 1 (those at the bottom of the stack).

Example

The capacity of the bay is 9 slots (3 tiers and 3 stacks). We have 7 containers. We assume that the order of retrieval is: 2/5/1/3/4/6/7. To retrieve the container number 2 we need firstly to lift-up the containers 6 and 4 (**Figure A**), and then put them on top of container (3), after that it is possible to lift up container (2).

In this example, the stages followed for container retrieval adhere to general principles and do not involve any heuristics.

Table A Total number of movements

Container number	2	5	1	3	4	6	7	
Number of reshuffles	2	2	1	2	1	0	0	
Total number of movements	3	3	2	3	2	1	1	**15**

The number of reshuffle is equal to 0 if only the total number of movements is equal to the total number of containers.

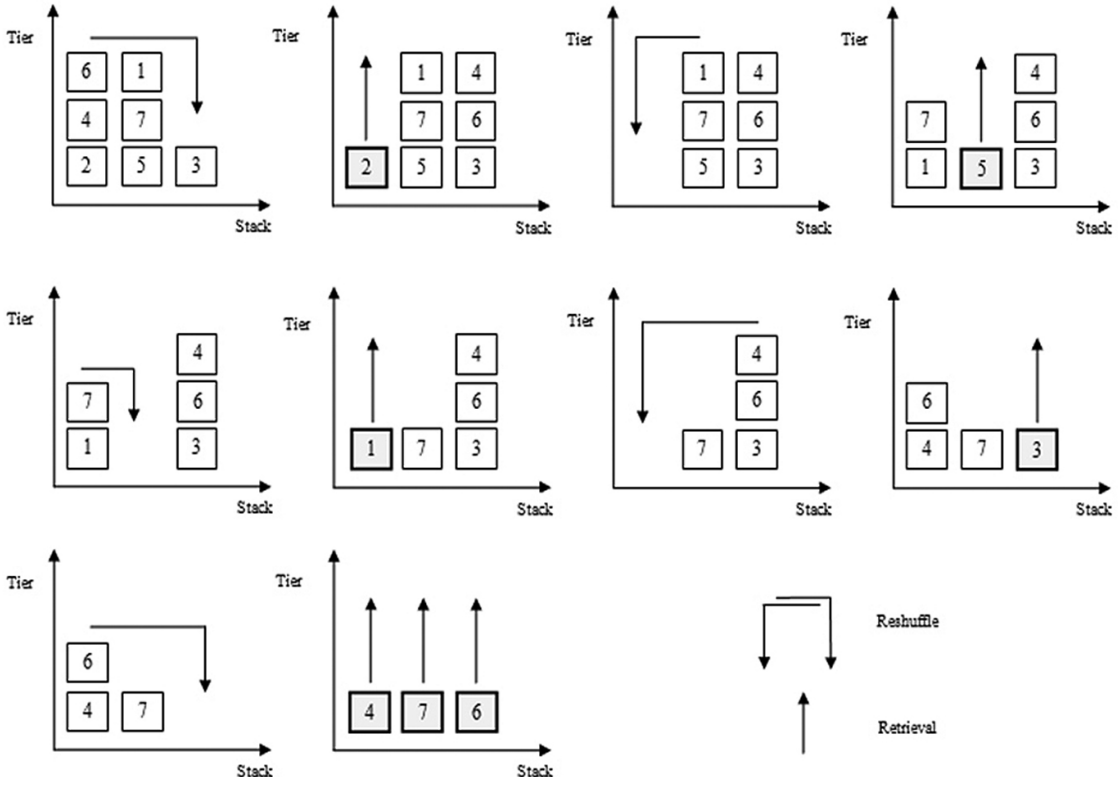

Figure A the process of container retrieval

References

[1] Ursavas, E.: Priority control of berth allocation problem in container terminals. Ann Oper Res (2015). DOI 10.1007/s10479-015-1912-7.

[2] Rodrigue, J., Notteboom, T.: Containerization, Box Logistics and Global Supply Chains: The Integration of Ports and Liner Shipping Networks. Maritime Economics & Logistics, 10, 152–174, 7 March 2008, 10.1057/palgrave.mel.9100196.

[3] Gharehgozli, A., Mileski, J., Duru, O.: Heuristic estimation of container stacking and reshuffling operations under the containership delay factor and mega-ship challenge, Maritime Policy & Management, 44:3, 373–391, DOI: 10.1080/03088839.2017.1295328.

[4] Meidute, L.: Comparative analysis of the definitions of logistics centres, Transport, 20;3:106–110. DOI: 10.1080/16484142.2005.9638005.

[5] Gharehgozli, H., Roy, D., Koster, R.: Sea container terminals: New technologies and OR models. 2015 Macmillan Publishers Ltd. 1479-2931 Maritime Economics & Logistics 1–38.

[6] Blažina, A., Ivče, R., Mohović, D., Mohović R.: Analysis of empty container management .Scientific Journal of Maritime Research 36 (2022) 305-317 © Faculty of Maritime Studies Rijeka, 2022. https://doi.org/10.31217/p.36.2.14.

[7] Gulić, M., Maglić, L., Valčić, S.: Nature Inspired Metaheuristics for Optimizing Problems at a Container Terminal. Scientific Journal of Maritime Research 32 (2018) 10-20 © Faculty of Maritime Studies Rijeka, 2018. https://doi.org/10.31217/p.32.1.16.

[8] SERBAN, C., CARP, D.: A Genetic Algorithm for Solving a Container Storage Problem Using a Residence Time Strategy. Studies in Informatics and Control, Vol. 26, No. 1, March 2017.

[9] Tang, L., Zhao, J., Liu, J.: Modeling and solution of the joint quay crane and truck scheduling Problem. European Journal of Operational Research 236 (2014) 978–990. http://dx.doi.org/10.1016/j.ejor.2013.08.050.

[10] Güven, C., and Eliiyi, D.: Modelling and optimisation of online container stacking with operational constraints, Maritime Policy & Management. DOI: 10.1080/03088839.2018.1450529.

[11] Ines Rekik, I., Elkosantini, S.: A multi agent system for the online container stacking in seaport terminals. Journal of Computational Science 35 (2019) 12–24. https://doi.org/10.1016/j.jocs.2019.06.003.

[12] Lee, Y., Hsu, N.: An optimization model for the container pre-marshalling problem. Computers & Operations Research 34 (2007) 3295–3313. doi: 10.1016/j.cor.2005.12.006.

[13] Bortfeldt, A., Forster, F.: A tree search procedure for the container pre-marshalling problem. European Journal of Operational Research 217 (2012) 531–540. DOI: 10.1016/j.ejor.2011.10.005.

[14] Izquierdo, C., Batista, B., Vega, M.: Pre-Marshalling Problem: Heuristic solution method and instances generator.

Expert Systems with Applications 39 (2012) 8337–8349. DOI: 10.1016/j.eswa.2012.01.187.

[15] Huang, S., Lin, T.: Heuristic algorithms for container pre-marshalling problems. Computers & Industrial Engineering 62 (2012) 13–20. DOI: 10.1016/j.cie.2011.08.010.

[16] Mohamed Gheith, Amr B. Eltawil & Nermine A. Harraz (2015): Solving the container pre-marshalling problem using variable length genetic algorithms, Engineering Optimization. DOI: 10.1080/0305215X.2015.1031661.

[17] Tierney, K., Pacino, D., Voß, S.: Solving the pre-marshalling problem to optimality with A* and IDA*. Flex Serv Manuf J. DOI 10.1007/s10696-016-9246-6.

[18] Torres, C., Valdes, R., Ruiz, R., Tierney, K.: Minimizing crane times in pre-marshalling problems. Transportation Research Part E 137 (2020) 101917. https://doi.org/10.1016/j.tre.2020.101917.

[19] Boge, S., Knust, S.: The parallel stack loading problem minimizing the number of reshuffles in the retrieval stage. European Journal of Operational Research 280 (2020) 940–952. https://doi.org/10.1016/j.ejor.2019.08.005.

[20] Kim, K., Hong, G.: A heuristic rule for relocating blocks. Computers & Operations Research 33 (2006) 940–954. doi: 10.1016/j.cor.2004.08.005.

[21] Ünlüyurt, T., Aydın, C.: Improved rehandling strategies for the container retrieval process. JOURNAL OF ADVANCED TRANSPORTATION 2012; 46:378–393. DOI: 10.1002/atr.1193.

[22] Chenhaoa, Z., Wencheng, W., Li Haobina, L.: Container reshuffling considered space allocation problem in container terminals. Transportation Research Part E 136 (2020) 101869. https://doi.org/10.1016/j.tre.2020.101869.

[23] Galle, V., Barnhart, C., Jaillet, P.: Yard Crane Scheduling for container storage, retrieval, and relocation. European Journal of Operational Research 0 0 0 (2018) 1–29. https://doi.org/10.1016/j.ejor.2018.05.007.

[24] Tan, C., He, J., Wang, Y.: Storage yard management based on flexible yard template in container terminal. Advanced Engineering Informatics 34 (2017) 101–113. http://dx.doi.org/10.1016/j.aei.2017.10.003.

[25] Xu, Y., Wang, M., Lai, K.K.; Ram, B. A Stochastic Model for Shipping Container Terminal Storage Management. Journal of Marine Science and Engineering (2022). https://doi.org/10.3390/ jmse10101429.

[26] Boschma, R., Mes, M., Vries, L.: Approximate dynamic programming for container stacking. European Journal of Operational Research 310 (2023) 328–342. https://doi.org/10.1016/j.ejor.2023.02.034.

[27] Lin, D &Chiang, C.: The Storage Space Allocation Problem at a Container Terminal, Maritime Policy & Management. DOI:10.1080/03088839.2017.1335897.

[28] CASTILHO, B and DAGANZO, C.: HANDLING STRATEGIES FOR IMPORT CONTAINERS AT MARINE TERMINALS. Transpn.Res.B.Vol.No. 2, pp. 151–166, 1993.

[29] Kim, K., Kim, H.: Segregating space allocation models for container inventories in port container terminals. International Journal of Production Economics 59 (1999) 415–423. PII: S 0 9 2 5 - 5 2 7 3 (9 8) 0 0 0 2 8 – 0.

[30] Saurí, S., Martín, E.: Space allocating strategies for improving import yard performance at marine terminals. Transportation Research Part E 47 (2011) 1038–1057. doi:10.1016/j.tre.2011.04.005.

[31] Armas, L., Valdes, D., Morell, C., Bello, R.: Solutions to Storage Spaces Allocation Problem for Import Containers by Exact and Heuristic Methods. Computación y Sistemas, Vol. 23, No. 1, 2019, pp. 197–211. ISSN 1405-5546. doi: 10.13053/CyS-23-1-2916.

[32] Hu, X., Liang, C., Chang, D., Zhang, Y.: Container storage space assignment problem in two terminals with the consideration of yard sharing. Advanced Engineering Informatics 47 (2021) 101224. https://doi.org/10.1016/j.aei.2020.101224.

[33] Mitchell, M.: Genetic Algorithms: An Overview. Complexity, 1 (1) 31–39, 1995.

[34] Katoch, S., Chauhan, S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications (2021) 80:8091–8126. https://doi.org/10.1007/s11042-020-10139-6.

[35] FORREST, S.: Genetic Algorithms. ACM Computing Surveys, Vol. 28, No. 1, March 1996.

[36] Holland, J.: Genetic Algorithms. Scientific American, Vol. 267, No. 1 (July 1992), pp. 66–73. http://www.jstor.org/stable/24939139.

[37] Kramer, O.: Genetic Algorithm Essentials, Studies in Computational Intelligence 679. DOI: 10.1007/978-3-319-52156-5_2.

[38] Lambora, A., Gupta, K., Chopra, K.: Genetic Algorithm- A Literature Review. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con), India, 14th – 16th Feb 2019.

[39] Min, H.: Genetic algorithm for supply chain modelling: basic concepts and applications. Int. J. Services and Operations Management, Vol. 22, No. 2, pp. 143–164.

[40] JAUHAR, S and PANT, M.: Genetic algorithms in supply chain management: A critical analysis of the literature. Sadhana Vol. 41, No. 9, September 2016, pp. 993–1017. DOI: 10.1007/s12046-016-0538-z.

[41] Lu, P., Wu, M., Tan, H., Peng, Y., Chen, C.: A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. J Intell Manuf. DOI: 10.1007/s10845-015-1083-z.

[42] Wu, M., Lin, C., Lin, C., Chen, C.: Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems. Computers and Operations Research 80 (2017) 101–112. http://dx.doi.org/10.1016/j.cor.2016.11.021.

[43] Lin, C., Lee, L., Wu, M.: Merits of using chromosome representations and shadow chromosomes in genetic algorithms for solving scheduling problems. Robotics and Computer Integrated Manufacturing 58 (2019) 196–207. https://doi.org/10.1016/j.rcim.2019.01.005.

[44] Hassanat, A., Prasath, V., Abbadi, M. Abu-Qdari, S., Faris, H.: An Improved Genetic Algorithm with a New Initialization Mechanism Based on Regression Techniques. Information 2018, 9, 167. doi: 10.3390/info9070167.

[45] Razali, N., Geraghty, J.: Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, 2011, London, U.K.

[46] Miller, B and Goldberg, D.: Genetic algorithms, tournament selection, and the effects of noise. Complex Systems, 9(3), 193–212 (1995).

[47] YU, F., FU, X., LI, H., DONG, G.: Improved Roulette Wheel Selection-Based Genetic Algorithm for TSP. 2016 International Conference on Network and Information Systems for Computers.

[48] Kumar, R.: Blending roulette wheel selection & rank selection in genetic algorithms. International Journal of Machine Learning and Computing, 2(4), 365 (2012).

[49] Wu, X., Wu, S.: An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem. J Intell Manuf. DOI: 10.1007/s10845-015-1060-6 (2015).

[50] Mirjalili, S.: Evolutionary Algorithms and Neural Networks, Studies in Computational Intelligence 780. https://doi.org/10.1007/978-3-319-93025-1_4.

[51] Zainuddin, F., Abd Samad, M.: A Review of Crossover Methods and Problem Representation of Genetic Algorithm in Recent Engineering Applications. International Journal of Advanced Science and Technology, Vol. 29, No. 6s, pp. 759–769 (2020).

[52] Murat Albayrak, M., Allahverdi, N.: Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms. Expert Systems with Applications 38 (2011) 1313–1320. doi: 10.1016/j.eswa.2010.07.006.

[53] Johnson, D., Papadimitriou, C., Yannakakis, M.: How easy is local search? Journal of computers and system sciences 37, 79–100 (1988).

[54] Aarts, E., van Laarhoven, P., Lenstra, J., Ulder, N.: A Computational Study of Local Search Algorithms for Job Shop Scheduling. ORSA Journal on Computing 6(2): 118–125. http://dx.doi.org/10.1287/ijoc.6.2.118.

[55] OMBUKI, B., VENTRESCA, M.: Local Search Genetic Algorithms for the Job Shop Scheduling Problem. Applied Intelligence 21, 99–109, 2004.

[56] Pongchairerks, P.: Efficient local search algorithms for job-shop scheduling problems. Int. J. Mathematics in Operational Research, Vol. 9, No. 2, pp. 258–277 (2016).

[57] Tseng, L., Lin, Y.: A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 198 (2009) 84–92. doi: 10.1016/j.ejor.2008.08.023.

[58] Ishibuchi, H., Murata, T.: A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, Vol. 28, No. 3, August 1998.

[59] OSOGAMI, T., OKANO, H.: Local Search Algorithms for the Bin Packing Problem and Their Relationships to Various Construction Heuristics. Journal of Heuristics, 9: 29–49, 2003.

[60] Levine, J., Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems. Journal of the Operational Research Society (2004) 55, 705–716. DOI: 10.1057/palgrave.jors.2601771.

[61] Yesil, C., Turkyilmaz, H., Korkmaz, E.: A New Hybrid Local Search Algorithm on Bin Packing Problem. 978-1-4673-5116-4/12/$31.00_c 2012 IEEE.

[62] Masson, R., Thibaut Vidal, T., Michallet, J., Penna, P., Petrucci, V., Subramanian, A., Dubedout, H.: An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. Expert Systems with Applications 40 (2013) 5266–5275. http://dx.doi.org/10.1016/j.eswa.2013.03.037.