sustainability

Article

Planning an Integrated Stockyard–Port System for Smart Iron Ore Supply Chains via VND Optimization

Álvaro D. O. Lopes [1], Helder R. O. Rocha [1,*,†], Marcos W. J. Servare Junior [1,2], Renato E. N. Moraes [2], Jair A. L. Silva [1,3] and José L. F. Salles [1,*,†]

1 Department of Electrical Engineering, Federal University of Espírito Santo, Av. Fernando Ferrari, 514, Vitória 29075-910, ES, Brazil; oliveira.alvaro4@gmail.com (Á.D.O.L.); marcos.servare@ufes.br (M.W.J.S.J.); jair.silva@ufes.br (J.A.L.S.)

2 Department of Industrial Engineering, Federal University of Espírito Santo, Av. Fernando Ferrari, 514, Vitória 29075-910, ES, Brazil; renato.moraes@ufes.br

3 Sensors and Smart Systems Group, Institute of Engineering, Hanze University of Applied Sciences, 9747 AS Groningen, The Netherlands

* Correspondence: helder.rocha@ufes.br (H.R.O.R.); jose.salles@ufes.br (J.L.F.S.)

† These authors contributed equally to this work.

Abstract: Stockyard–port planning is a complex combinatorial problem that has been studied primarily through simulation or optimization techniques. However, due to its classification as non-deterministic polynomial-time hard (NP-hard), the generation of optimal or near-optimal solutions in real time requires optimization techniques based on heuristics or metaheuristics. This paper proposes a deterministic simulation and a meta-heuristic algorithm to address the stockyard–port planning problem, with the aim of reducing the time that ships spend in berths. The proposed algorithm is based on the ore handling operations in a real stockyard–port terminal, considering the interaction of large physical equipment and information about the production processes. The stockyard–port system is represented by a graph in order to define ship priorities for planning and generation of an initial solution through a deterministic simulation. Subsequently, the Variable Neighborhood Descent (VND) meta-heuristic is used to improve the initial solution. The convergence time of VND ranged from 1 to 190 s, with the total number of ships served in the berths varying from 10 to 1000 units, and the number of stockyards and berths varying from 11 to 15 and 3 to 5, respectively. Simulation results demonstrate the efficiency of the proposed algorithm in determining the best allocation of stockpiles, berths, car-dumpers, and conveyor belts. The results also show that increasing the number of conveyor belts is an important strategy that decreases environmental impacts due to exposure of the raw material to the atmosphere, while also increasing the stockyard–port productivity. This positive impact is greater when the number yards and ship berths increases. The proposed algorithm enables real-time decision-making from small and large instances, and its implementation in an iron ore stockyard–port that uses Industry 4.0 principles is suitable.

Keywords: stockyard–port allocation system; iron ore; deterministic simulation; Variable Neighborhood Descent

check for updates

Citation: Lopes, Á.D.O.; Rocha, H.R.O.; Servare Junior, M.W.J.; Moraes, R.E.N.; Silva, J.A.L.; Salles, J.L.F. Planning an Integrated Stockyard–Port System for Smart Iron Ore Supply Chains via VND Optimization. *Sustainability* **2023**, *15*, 8970. https://doi.org/10.3390/su15118970

Academic Editors: Alireza Daneshkhah and Maryam Farsi

Received: 23 March 2023
Revised: 3 May 2023
Accepted: 29 May 2023
Published: 1 June 2023

1. Introduction

The most competitive logistic companies are those whose supply chains adapt to changes with the greatest speed and flexibility [1]. To effectively deal with a competitive market, supply chains are adopting the Industry 4.0 (I4.0) concept using the main technological innovations in the fields of automation, control, and information technology in manufacturing processes. From connected systems like cyber physics, Internet of Things, and Internet of Services, supply chains tend to become even more efficient, autonomous, and personalized [2,3]. Smart supply chains are characterized by optimal decision-making based on mathematical analysis of data, artificial intelligence, machine learning, and simulation-based approaches [4,5]. According to Ferreira et al. [6], there has been an increasing trend in the number of publications on simulation in I4.0. They suggest that digital twin and hybrid simulations, which represent a combination of optimization and computer simulations, are the most promising approaches for I4.0 due to their ability to capture principles related to integration, real-time capability, flexibility, real-time solutions, autonomy, and optimization [7,8]. Indeed, optimization and hybrid simulation models can be used to assist in decision-making quickly before implementation in real supply chains [6]. A natural extension in the use of hybrid simulation models is to evaluate several alternative scenarios in order to obtain the best performance of the supply chain [9]. In addition to the economic benefits, optimization can also help reduce environmental impacts, energy costs, and violations of regulations, and can assist in the most efficient way to deal with uncertainties in both production and customer orders [10].

However, the mining industry supply chain has critical strategic issues involving railways, ports, and long-distance maritime shipping [11]. Before reaching a cargo ship, the mining materials traverse stockyards with well-planned operations in order to avoid impairments in ship delivery. This means that a stockyard represents a bottleneck in the chain and significantly influences the mining logistic chain performance [12]. The main processes considered in stockyard–port planning are the arrival of the wagons to the dumpers, the routes that help in the stacking and recovery of ore, the stacking and recovery process, and the loading of ships. Therefore, the operational efficiency of the stockyard–port requires that the flow of ore in each process is carried out quickly and without interruption, at the same time as the ships are loaded with the quantity and quality of ore requested by customers. The implementation of efficient logistic practices and inventory management strategies in the stockyard–port planning, utilizing optimization techniques, can reduce dock time for ships, increase the operational efficiency of ports, and decrease the environmental impacts due to the transport time reduction of the raw material exposed to the atmosphere. This, in turn, can positively impact the carbon footprint of freight transport. Furthermore, the integration of I4.0 principles can contribute to sustainability efforts by reducing resource and energy consumption, improving the transparency and efficiency of industrial processes.

1.1. Related Work

Several studies related to stockyard–port planning using deterministic optimization techniques have been proposed in the last decade. Ago et al. [13] proposed a Mixed-Integer Linear Programming (MILP) model with Lagrangian decomposition and coordination techniques to simultaneously optimize the storage allocation and transportation routing at the raw materials yard. Servare et al. [14] developed a MILP (see Table 1) model and a linear relaxation-based heuristic to minimize the energy costs of an iron ore stockyard, including the production of mines and the demands of the berth. The authors of [15,16] proposed integer programming models with greedy construction and enumeration to optimize stockyard management, whereas Hanoun et al. [17] formulated a bi-objective model for stockyard planning with resource scheduling. In [18,19], Belov et al. introduced a constraints programming model with a large neighborhood search that considers reclaimer scheduling and vessel arrivals.

Savelsbergh and Smith [20] proposed a tree search algorithm for stockyard planning that utilizes space-time diagram geometric properties of coal stockyard planning. Menezes el al. [21] presented a mathematical model to control the stockyard–port system, which includes receipt through a rail system, stockyard equipment allocation, and availability at the port, taking into account production quantities, prices, and demands. The proposed solution was based on column generation with branch and price. Moreno et al. [22] proposed several variants of MILP models to optimize a combined version of open-pit mine production scheduling with stockyard planning. Discrete event simulation models applied in the stockyard–port planning can be found in [23], in which the authors evaluated the value of different production plans with respect to some performance function of interest.

Xiao-ping et al. [24] proposed a simulation model of the conveyor network system of a surface mine, and van Vianen et al. [25] performed a discrete event simulation to redesign a conveyor belt network.

In [26], a metaheuristic optimization was presented for complex iron ore and coal chains with integrated operations from raw material extraction to the market. The optimization takes into account total revenue, cost, productivity, contractual penalties and bonuses, and energy consumption, considering the entire chain as a dynamic structure due to changes in commodity markets. Gao [27] studied the problem of sequencing ships based on the contract of a steel plant that receives iron ore, coal, and stone. The model was formulated as a mixed integer scheduling problem and solved by column generation, minimizing the total demurrage costs within a given planning time horizon. The results were compared with CPLEX to verify the effectiveness of the methodology. In [28], storage space was allocated in a yard considering unloading, stacking, and recovery operations, through a Mixed Integer Problem (MIP) formulation. An approach based on Benders decomposition was applied to decompose the problem and solve it efficiently and quickly.

Silva et al. [29] addressed the problem of mixing stocks in a nickel mine, formulating an optimization problem using MILP to minimize the total mass target, the deviations from grade targets, and the number of used stockpiles. In computational experiments, optimal solutions were found in a few seconds. Tang [30] investigated the allocation of storage space in stockyards at iron ore terminals with mixed ore. The MIP model minimized the total distance of iron ore displacement, and a genetic algorithm was also used to efficiently obtain near-optimal solutions. Computational experiments showed that large instances were solved almost optimally in just a few seconds. Finally, Leal et al. [11] presented a review of operational research in the mining industry. They described applications isolated from the chain first and thereafter connecting subsystems of the mining chain. They indicated that stochastic models involving environmental issues and artificial intelligence can be used to solve more complex problems.

Based on the presented literature review, it is clear that the stockyard–port planning problem is a complex combinatorial problem that has been studied primarily through mathematical programming techniques with limitations related to the size of data instances. Thus, we identified that the gap to be filled is to solve large input instances while considering a long planning horizon in a few seconds. In this article, we propose the combination of a simulation tool for stockyard–port operations with a VND metaheuristic to efficiently allocate stacks and ships in the port's berth, minimizing the total time of equipment operations in the stockyard, from train cars unloading at the terminal to ore loading onto the ships.

Table 1. Acronyms and programming techniques (important related references are also shown).

Acronym	Explanation	Ref.
CB	Conveyor Belt	-
CD	Car Dumper	-
IOSpT	Iron Ore Stockyard-Planning Terminal	[14]
MILP	Mixed-Integer Linear Programming	[13,14,29]
MIP	Mixed-Integer Programming	[28,30]
RE	Reclaimer	-
SR	Stacker–Reclaimer	-
ST	Stacker	-
SY	Stockyard	-
SB	Ship Berth	-
VND	Variable Neighborhood Descent	[31], this work

1.2. Formulation of the Iron Ore Stockyard Planning Terminal Problem

The main goal of the work described in this paper is to solve the problem of stockpile allocation in a stockyard–port with general characteristics. The problem consists of optimizing the configuration of the stockyard and berth equipment and choosing the routes for transporting iron ore and for determining the positions for the ore piles in the stockyard. Additionally, it includes the definition of the berths for ships in order to minimize the total time spent on stockyard operations, such as stacking and reclaiming processes of ore piles and their loading on ships. Therefore, the optimization problem was formulated as follows:

- **Objective Function**: minimizing the elapsed time between the port's first ship arrival and the completion of the transshipment operation of the last ship leaving the port;
- **Constraints**: the lengths and velocity of the stacking and reclaiming conveyor belts, and the number and volume of stockyards used for stacking and reclaiming.

1.3. Contributions

As contributions of this work, we highlight:

- The design of abstracted graph representations to capture the main concepts of a real stockyard–port system;
- The graph representation that allows a robust and effective implementation of a VND metaheuristic that optimizes the stockyard–port NP-hard planning problem;
- The proposal of a novel, flexible, and fast Deterministic Simulation Algorithm (DSA) to generate solutions and calculate the objective function;
- A description of a hybrid method combining the VND metaheuristic with the DSA. The hybrid method was used to determine the minimum total time spent to complete the stockyard–port planning process;
- The application of the hybrid approach to analyze a real stockyard–port system, using real data from its conveyor belts, stockpiles, berths, and ships.

The remainder of this paper is divided as follows. Section 2 provides the characteristics of the stockyard–port system, Sections 3 and 4 describe the simulation algorithm and the *VND* metaheuristic applied to an Iron Ore Stockyard-Planning Terminal (IOSpT). Sections 5 and 6 present the computational results and conclusions, respectively.

2. Features of the Stockyard–Port System

This section describes the functionality and features of the idealized IOSpT presented in Figure 1. The port is specifically dedicated to the exportation of iron ore and has three berths. The stockyard–port system comprises several subsystems, including an unloading terminal with five car dumpers (CDs), several pelletizing plants, stockyards for the inputs used in the pelletizing process, eleven stockyards (SYs) for the allocation of stockpiles, ship berths (SBs), and equipment associated with these processes, such as stackers (STs), reclaimers (REs), stackers–reclaimers (SRs), and shiploaders. The stacking equipment consists of nine units, of which three are hybrids (SRs that can stack and reclaim materials), eight are reclaimers, and three are the same hybrids used in the stacking process. The stackers interact with the car dumpers, with the conveyor belt (CB) that runs from the unloading terminal to the stockyards, as well as with the stockyards. The reclaimers serve the stockyards and the conveyor belt, where the stockpiles are reclaimed and transported from the stockyards to the piers. Stackers and reclaimers move to reach the piles in their positions in the stockyards.

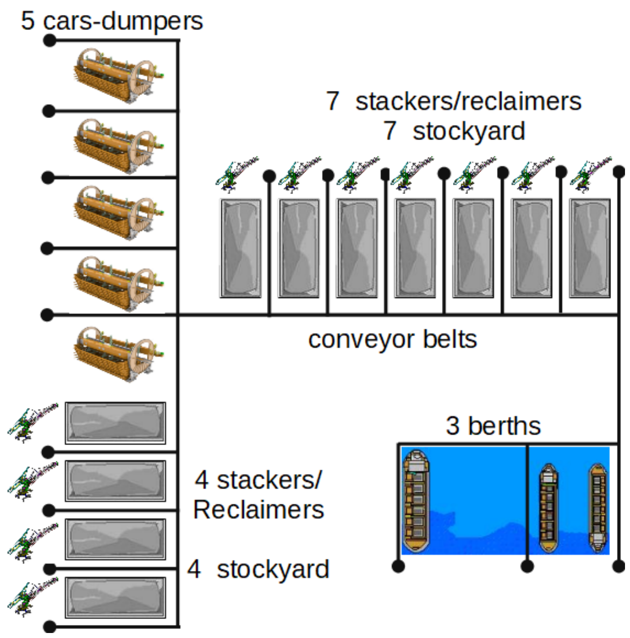

Figure 1. Graphical representation of the idealized Iron Ore Stockyard-Planning Terminal (IOSpT).

2.1. *Iron Ore Unloading Terminal*

At the beginning of the process, iron ore is transported from the mines to the terminal by rail cars towed by a locomotive. When the locomotives arrive at the unloading terminal, the wagons are unloaded by equipment called car-dumpers [32]. In the IOSpT, there are five car-dumpers (CD), and each one has a handling capacity of 8000 tons per hour. Each car-dumper receives one wagon at a time, unloading the ore onto a conveyor belt. The ore is then conveyed by belts to the stockyards, which are composed of eleven ore deposit areas with a total capacity of m^3 capable of storing up to 10.5 million tons [14].

2.2. *Stockyards*

Stockyards are spaces for storing stockpiles. There are 11 stockyards in the IOSpT divided into 2 areas, as shown in Figure 1. Stockpiles are stored on a first-come, first-served basis.

2.3. *Stockpiles*

Stockpiles correspond to the products stored in the stockyards and have different sizes. They can be handled using stackers and reclaimers. The material-handling equipment represents the conveyor belts with their deviations and connections, both for the stacking of the stockpiles and for their reclaiming. The handling material equipment comprises the station and the conveyor belts with their deviations and connections for the stacking of the stockpiles and for their reclaiming. Similar to the work described in [33], we consider two types of conveyor belts with different capacities. The daily operating period must be respected when a conveyor belt is used in order to avoid damage. Thus, a conveyor belt with a capacity of 8000 tons per hour, with 8 hours of work per day, supports an amount of 64,000 tons of ore per day.

2.4. *Routes*

Routes are paths (vectors of conveyor belts) that interconnect processes in the port logistics chain such as from car-dumpers to stockyards and from stockyards to berths.

2.5. *Stacking*

The operation of stacking the iron ore stockpile consists of adding a product to a particular stockpile or empty space that is carried out by a stacker. The products, transported afterward by the conveyor belts, are stacked, forming stockpiles in the stockyards. The stacker allocates iron ores since the conveyor belt is not at its daily working limit, or since the capacity of the stockyard location is not extrapolated. If the stacker works in more than one stockyard and one of the stockyards overflows, it allocates the rest of the products to another stockyard. If the other stockyard also extrapolates, the stacker waits until the stockyards and the conveyor belt are ready to proceed with the stacking.

2.6. Reclaiming

This consists of removing material or product from a specific stockpile made by bucket wheel reclaimers. SRs are also used to reclaim hybrid stockpiles. After the placement of the stockpiles in the stockyard, the ship is allowed to enter the berth. Once docked, the stockpile reclaimed for boarding is immediately calculated. The product is transported for boarding via routes from the stockyards to the berths. Reclaiming is also done by respecting the capacity of the conveyor belts transporting the products for shipment. If the conveyor belt reaches its daily working limit, it is disabled, and reclaiming is interrupted. Reclaiming is done stockpile by stockpile. The first reclaimed stockpile is the first to be loaded onto the ship. After the last stockpile is loaded on the ship, it is released from the berth. Then, historical information about this ship is obtained, such as the moment the stacking starts, the moment the ship berths, the beginning of the ship's reclaiming, and the departure of the ship. This history is used to calculate the total time specified in the objective function.

2.7. Berths

A ship may dock at the pier only after its respective iron ore stockpiles have been formed in the stockyard. The mooring at the pier has to be according to the ship's capacity. Therefore, before starting the reclaiming of the stockpiles for a ship, it must dock at a pier that has characteristics to support the relevant size of the ship.

2.8. Ships

The first step in planning a ship's boarding at a port is to appoint it, which consists of specifying which ship should arrive and when. After the appointment, the port's planning and programming teams already know which product should be made available for boarding the ship, which allows for an availability check of this product at the port and, eventually, makes it a transport priority along the entire route (from the loading of the cargo on the railroad, to the unloading of the wagons in the port, and then to the loading of the ships). With the ship's arrival at the port, a mooring order is issued [34]. After this operation, the docking and loading of the product follow. In this work, three types of ships were used: Panamax, Capesize, and Valemax . The type of vessel is defined by its product storage capacity in tons. The piers have capacities equivalent to those of the types of ships supported by the IOSpT. Therefore, we can state that, in the SB1, ships are moored up to the Capesize type, and in the SB2, they are moored up to the ultra-large crude carrier type, without extrapolating the 330,000 tons of ore; otherwise, they will dock at SB3.

3. The Developed Simulation Algorithm

In this work, we used programming techniques based on the graph representation [35] and queue concepts to implement the algorithm proposed to solve the iron ore stockyard planning and berth allocation in a port terminal. Simulation data were obtained according to the following procedures:

- Identify the technical reports with relevant information such as the characteristics and operational data of the stockyard–port system;
- Use electronic spreadsheets for data mining of technical reports;
- Validate the extracted data according to comparison with other sources of information, and verify consistency and precision;
- Generate data input files in the proper format to be used in the developed simulation software;

- Compare the data obtained in the simulation with the information extracted from the technical report to validate the model;
- Perform additional analysis to identify trends or patterns in the simulation results.

In the following, we describe the proposed deterministic simulation algorithm, as well as the VND deployed to improve their solutions.

3.1. The Proposed Deterministic Simulation Algorithm

Given the set of ships Qn with their demands and types; the set of berths Qb; the set of stockyards Qp with their capacities; the set of conveyor belts Qe with their routes; the lengths, speed, and daily capacities; and the configuration of stockyards and berths CPB (stockyards available to stack and reclaim, and types of ships that can be moored in each berth), the DSA outputs the routes for transporting iron ore, the positions for the ore piles in the stockyard, the moment the stacking starts, the moment the ship berths, the beginning of the ship's reclaiming, and the departure of the ship such that all ship demands are satisfied and the conveyor belts' and stockyards' capacities and availability are respected.

The sequence of ships' demands $Dm(Qn)$ defines the amount of ore ordered by each vessel. The amount of ore ordered also defines the vessel type. If an order has 390,000 tons of ores, it is a Valemax type ship, referred to as *string* "Va". If an order contains 180,000 tons of ore, it is a Capesize type vessel, referred to as *string* "Ca". The same reasoning goes for the Panamax type ship, referred to as *string* "Pa". The algorithm handles the demands in an arbitrary order given by the input file; that is, the algorithm builds a queue of ships according to how the set of ships appears in the input file instance. The algorithm applies the queue First In–First Out (FIFO) method, where the first ship's demand in the queue is the first to be loaded. For each ship, the algorithm runs when assigning the routes for transporting the iron ore demanded, the positions to place the iron ore piles in the stockyard, the moment the stacking starts, the moment the ship berths, the beginning of the ship reclaiming, and the moment ship departs. The configuration and restrictions of the conveyor belts and stockyards define the transportation routes, stockyard positions, and the moment of stacking. The moment of ship berths is set immediately after all the iron ore of a ship is allocated in the stockyard. After ship mooring, the reclaiming is allowed to begin, and the operation performs until the shipment is finished. A ship can depart immediately after the last pile is recovered.

In the DSA, a fixed and constant number of five car-dumpers is used in the unloading terminal, which receives the wagons of the train one at a time, unloading them on a conveyor belt. The discharge follows the order of the stockpile as defined by the algorithm. The algorithm also considers a fixed capacity of 10,000 tons for stockpiles. Each train carries a number of wagons equivalent to 1 stockpile, that is, 10,000 tons.

The other physical means to be taken into account for the port system are conveyor belts. The conveyor belts connect the car-dumpers with the stockyards and the stockyards with the berths, forming routes for the traffic of the stockpiles. Therefore, we create a graph representing physical means and distances between paths. In the graph, the turners, stockyards, and berths are the vertices, and the edges are the conveyor belts. Due to the scarcity of information, the lengths of the conveyor belts in the input instances are calculated randomly through a normal distribution with an average of 1000 m and standard deviation of 500 m. For the belts, we stipulate an average speed of around 32 Km/h. Thus, the transportation time (T) is calculated by dividing the length (S) by the speed (V). Knowing the length of the conveyor belts and their speed, the DSA creates a weighted adjacency matrix filled with the transportation times in hours (h) from a stockpile for each conveyor belt, as shown in Table 2. Table 2 shows an example of the above-mentioned matrix, considering a terminal with one set of CD, 11 stockyards (see Figure 1) represented by the letters A, B, C, D, E, F, G, H, I, J, and K, and 3 berths denoted as SB1, SB2, and SB3.

Table 2. Weighted matrix of stacking and reclaiming times (h).

	CD	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**SB1**	**SB2**	**SB3**
CD	0	0.025	0.046	0.025	0.024	0.045	0.02	0.046	0.046	0.045	0.046	0.038	0	0	0
A	0.025	0	0	0	0	0	0	0	0	0	0	0	0.024	0.024	0.024
B	0.046	0	0	0	0	0	0	0	0	0	0	0	0.027	0.027	0.027
C	0.025	0	0	0	0	0	0	0	0	0	0	0	0.050	0.050	0.050
D	0.024	0	0	0	0	0	0	0	0	0	0	0	0.031	0.031	0.031
E	0.045	0	0	0	0	0	0	0	0	0	0	0	0.030	0.030	0.030
F	0.02	0	0	0	0	0	0	0	0	0	0	0	0.034	0.034	0.034
G	0.046	0	0	0	0	0	0	0	0	0	0	0	0.053	0.053	0.053
H	0.046	0	0	0	0	0	0	0	0	0	0	0	0.022	0.022	0.022
I	0.045	0	0	0	0	0	0	0	0	0	0	0	0.016	0.016	0.016
J	0.046	0	0	0	0	0	0	0	0	0	0	0	0.051	0.051	0.051
K	0.038	0	0	0	0	0	0	0	0	0	0	0	0.025	0.025	0.025
SB1	0	0.024	0.027	0.050	0.031	0.030	0.034	0.053	0.022	0.016	0.051	0.025	0	0	0
SB2	0	0.024	0.027	0.050	0.031	0.030	0.034	0.053	0.022	0.016	0.051	0.025	0	0	0
SB3	0	0.024	0.027	0.050	0.031	0.030	0.034	0.053	0.022	0.016	0.051	0.025	0	0	0

At port terminals, conveyor belts differ in terms of daily work capacities. In the proposed algorithm, the conveyor belt that reaches its limit is disabled. As an illustrative example, we can consider a case in which conveyor belts support 8000 tons per hour and others 16,000. The conveyor belts with a capacity of 8000 tons per hour support an amount of 80,000 tons of ore, with 10 h of work per day. In the input instances files, capacities are distributed at random due to a lack of related information. Knowing the conveyor belt parameters, the DSA defines the routes for stacking and reclaiming the stockpiles. Stackers allocate stockpiles one at a time according to the order of the queue of ships, respecting the conveyor belt's daily capacity and the stockpile capacity. Through the variables described above, the times related to the operations of the conveyor belts from the car-dumpers to the stockyards and from the stockyards to the berths are well defined. Consequently, we can plan the total time spent in the terminal, from the first ship to the last.

After all ships' stockpiles are formed in the stockyard, the ship docks at the berth and begins reclaiming from the related initial stockpile as placed in the stockyard. The stockpile is loaded onto the ship immediately after being reclaimed, regardless of the loading method. The ship is unberthed immediately after the last pile is recovered. In the DSA, there are some rules for the end of the day, as described in Table 3, where: "NA" means not available, "c" means continue with the planning, and "r/e" stands for ending the day and updating the positions in the matrix of the reclaiming and stacking belts' capacities to the values at the beginning of the system.

The rules consider several situations regarding the capacity of conveyor belts and ships to carry out the planning. For example, the end of the working day (r/e) occurs when a ship is able to enter the berth (Ships = 1), the conveyor belts from the car-dumpers to the stockyards are disabled (CR/SY = 0), conveyor belts from stockyards to berths are available (SY/SB = 1), and berths cannot receive ships (berths = 0). In this example, an impasse is reached, being unable to stack or reclaim. Thus, the working day is over.

The strategy used to implement the minimization of the objective function is to systematically change configurations of the stockyards and berths using the proposed VND metaheuristic. These configurations affect how the deterministic simulation algorithm assembles the solution to the problem and, consequently, changes the value of the objective function. Therefore, the DSA requires one more input, with the configuration settings of the current stockyards and berths represented as a vector named CPB. The CPB is obtained by joining a stockyard configuration vector and a berth configuration vector. The stockyard configuration vector indicates which stockyards are available to stack and reclaim stockpiles. The berth configuration vector defines which type of ship can be moored in each berth.

Table 3. Daily rules.

Ships	CB (CR to SY)	CB (SY to SB)	Berths	Output
0	0	0	0	NA
0	0	0	1	r/e
0	0	1	0	NA
0	0	1	1	c
0	1	0	0	NA
0	1	0	1	r/e
0	1	1	0	c
0	1	1	1	c
1	0	0	0	r/e
1	0	0	1	r/e
1	0	1	0	r/e
1	0	1	1	c
1	1	0	0	c
1	1	0	1	c
1	1	1	0	c
1	1	1	1	c

The stockyard configuration vector indicates which stockyards are available to be occupied by the iron ore stockpiles. As an example, consider a terminal with 11 stockyards (see Figure 1) represented by the letters A, B, C, D, E, F, G, H, I, J, and K, as shown in Table 4. The first position of this vector represents stockyard A, the second one represents stockyard B, and so on. Each element of this vector can be set with a binary value. It is set to 0 to indicate that the respective stockyard is not available to be used in the planning; otherwise, it is set to 1. In Table 4, stockyards A, B, D, E, F, H, and K are available to be used. Stockyards C, G, I, and J are unavailable, which means that no stacking or reclaiming operations will be carried out in these stockyards.

Table 4. Stockyard configuration.

A	B	C	D	E	F	G	H	I	J	K
1	1	0	1	1	1	0	1	0	0	1

The berth configuration vector defines which type of ship can be moored in each berth. The algorithm considers that more than one type of ship can be moored in each berth. The berth occupation is represented by a sequence of 3 binary digits characterizing 3 types of ships, the combination of which results in 8 mooring possibilities. Table 5 shows the configuration for a berth. If a given berth has the bit configuration "001", only Panamax vessels are allowed to dock. If the configuration of the same berth is "111", docking is allowed for Panamax, Capesize, and Valemax vessels, and so on. It can be seen from Table 5 that Capesize vessels are allowed to dock in berth 1, Panamax vessels in berth 2, and Capesize and Valemax vessels in berth 3. Once the stockyard and berth configurations are defined, the stockyard vector and berth configurations are obtained, as shown in Table 6.

Table 5. Berth configuration.

0	0	0	maintenance
0	0	1	“Pa”
0	1	0	“Ca”
0	1	1	“Va”
1	0	0	“Pa” and “Ca”
1	0	1	“Pa” and “Va”
1	1	0	“Va” and “Ca”
1	1	1	“Pa”, “Ca”, and “Va”

Table 6. Stockyard and berth configuration.

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**SB 1**	**SB 2**	**SB 3**
1	1	0	1	1	1	0	1	0	0	1	0 1 0	0 0 1	1 1 0

3.2. *Pseudocode of the Deterministic Simulation Algorithm*

The DSA presented in Algorithm 1 takes as inputs Qn, Qb, Qp, and Qe, as well as lengths, speed, daily capacities, and the configuration of CPB. All used input instances ensure that the total capacity of the stockyards is sufficient to meet all demands. However, during the planning process, a specific stockyard or the conveyor belts that serve it may have their capacity exhausted at a given moment. The nomenclatures of the DSA are presented in Table 7.

Table 7. Variables and parameters of the Algorithm.

Nomenclature	**Description**
Qn	set of ships and their attributes
$Dm(Qn)$	demand of each ship
Qb	set of berths
Qp	set of stockyards and their attributes
Qe	set of conveyor belts and their attributes
CPB	binary set with the availability configuration of stockyards and berths
CEs	binary set with the availability configuration of stacking conveyor belts
CEr	binary set with the availability configuration of reclaiming conveyor belts
MW	weighted matrix of stacking and reclaiming times
CD	current demand
CS	current stockyard
$TimeS$	time involved in the current stacking operation
$TimeD$	time involved in the current docking operation
$TimeR$	time involved in the current reclaiming operation
$TimeU$	time involved in the current unberthing operation
H	planning time (objective function)

Algorithm 1 creates the weighted matrix MW of stacking and reclaiming times in line 1. It also creates the auxiliary data structures CEs and CEr (line 2) as a vector of ones to monitor the availability of the stacking and reclaiming conveyor belts, respectively. Each conveyor belt has its position in the vector as 1 if available and 0 otherwise. In line 3, the status of the set of variables $DailyRules$ (see Table 2) is set to initial valid states, i.e, no ship is able to enter the berth ($Ships$ = 0). It is worth mentioning that, at this stage, there is at least one enabled conveyor belt from the car-dumpers to the stockyards ($CB\_CRSY = 1$), one available conveyor belt from the stockyards to the berths ($CB\_SYSB = 1$), and at least one berth to receive ships ($Berths$ = 1). In line 4, the objective function H is initialized as 0.

The simulation must attend to all requirements, which means that the algorithm enters a loop to check each available stockyard regarding the sequence of their demands $Dm(Qn)$ (loop from line 5 to 18), verifying which operation (stacking, docking, unberthing, and reclaiming) can be performed at that moment, given the current situation of all resources (stockyards, conveyor belts, berths, and ships) and the stockyard being checked (*CS*). For example, an empty stockyard cannot perform reclaiming but can perform stacking, and the docking operation can only be performed immediately after all the iron ore of a ship is allocated in the stockyard. The reclaiming operation can only be performed if the stockpile in the current stockyard has its respective ship moored, and a ship can only be unberthed immediately after the last pile is recovered.

Algorithm 1: DSA(*Qn*, *Qb*, *Qp*, *Qe*, *CPB*)

```
1  $MW \leftarrow$ BuildRoutingTimes(Qe)
2  $CEs, CEr \leftarrow \mathbb{1}$
3  $DailyRules \leftarrow$ Set$(Qn, Qb, Qp, Qe, CPB)$
4  $H \leftarrow 0$
5  while $Dm(Qn) \neq \emptyset$ do
6      $CD \leftarrow$ Pop$(Dm(Qn))$
7      $CS \leftarrow$ NextAvailable$(Qp, CPB, CS)$
8      $DailyRules, CEs, CPB, TimeS \leftarrow$ Stacking$(CD, CS, CEs, MW)$
9      $DailyRules, Qn, Qb, TimeD \leftarrow$ Docking$(CS, Qn, Qb, CPB)$
10     $DailyRules, Qn, Qb, CEr, CPB, TimeR \leftarrow$ Reclaiming$(CS, Qn, Qb, CEr, MW)$
11     $DailyRule, Qn, Qb, TimeU \leftarrow$ Unberthing$(Qn, Qb)$
12     $H \leftarrow$ Update$(H, TimeS, TimeD, TimeR, TimeU)$
13     if FinishDay$(DailyRules) == TRUE$ then
14         $H \leftarrow$ Update$(H, nightshift)$
15         $CEs, CEr \leftarrow \mathbb{1}$
16         $DailyRules \leftarrow$ Reset$(Qn, Qb, Qp, Qe, CPB, DailyRules)$
17     end if
18 end while
19 Return H
```

In line 6, the next demand from the sequence of demands $Dm(Qn)$ is assigned if the current is removed from $Dm(Qn)$ using the `Pop()` function. Given the set of stockyards Qp, the set with the configuration of available stockyards CPB, and the current stockyard CS, the function `NextAvailable()` defines the next available stockyard CS to be processed (line 7). In line 8, the function `Stacking()` executes all the stacking operations in the current stockyard to attend current demand CD, considering the stacking conveyor belts' restrictions (CEs) and stacking times (MW). It returns the changes in $DailyRules$, conveyor belt restrictions (CEs), the availability of the current stockyard (if CS capacity has reached its limit, it must be disabled), and the times involved. In line 9, the function `Docking()` checks if there is any Qn ship that can be docked in any available berth (given by examination of Qb and CPB) to retrieve some stockpile from the current stock CS. If a docking operation is performed, it affects the $DailyRules$, the ships (Qn), and berth (Qb) status, and it is then necessary to compute the involved time ($TimeD$).

The functions `Reclaiming()` (line 10) and `Unberthing()` (line 11) execute, respectively, reclaiming and unberthing by checking all possible situations and returning the result of changes in $DailyRules$, involved times, resources capacities, availability, and status. The objective function H is then updated with the times involved in the previous operations in line 12. In line 13, the rules in the Table 3 are checked and, if true, the DSA ends the day updating, in line 14, the objective function with the time interval between the end of the day and the start of operations of the next day. Since the current day is over, the conveyor belt daily capacity (line 15) and the $DailyRules$ (line 16) are reset for the next day. The final planning time is returned in line 19.

Considering n as the number of ships in Qn, b the number of berths in Qb, p the number of stockyards in Qp, and e the number of conveyor belts in Qe, the DSA provides a main loop with time complexity in $O(n)$ (all demands meet). Inside the main loop, the function `NextAvailable()` runs in $O(p)$, function `Stacking()` runs in $O(e^2)$, function

`Docking()` runs in $O(n \times b)$, function `Reclaiming()` runs in $O(e^2)$, and function `Unberth()` runs in $O(n)$. All other functions and instructions run $O(1)$. Thus, Algorithm 1 runs with a time complexity given by

$$\begin{aligned} C_T &= O(n) \times [O(p) + O(e^2) + O(n \times b) + O(e^2) + O(n)] \\ &= O(np) + O(ne^2) + O(n^2 b) + O(ne^2) + O(n^2) \\ &= O(np) + O(ne^2) + O(n^2 b) + O(n^2) + O(nb). \end{aligned} \tag{1}$$

4. The Proposed *VND* Metaheuristic

Metaheuristics are convenient for NP-complete optimization problems for which there are no polynomial time solution algorithms or methods. To solve NP-complete problems efficiently, the following characteristics of the metaheuristic are adopted:

- Strategies to achieve an optimal solution through "efficient" search spaces with neighborhood structures;
- Usage of solutions found during the optimum search to generate other solutions;
- Intensification that seeks to find a great location and diversification that aims to leave the great location (disturbance).

The Variable Neighborhood Descent is a metaheuristic proposed by Mladenovic in 1997 [31]. It is designed to find approximate solutions and can be applied to large problems of linear programming, entire programming, non-linear programming, and others. VND is based on systematic changes in the neighborhood of the solutions to solve combinatorial optimization problems; whenever there is improvement in a certain neighborhood, it returns to the less distant neighbor [31]. The neighborhoods $N = \{\mathrm{N}^1, \mathrm{N}^2, \ldots, \mathrm{N}^{\mathrm{k_{max}}}\}$ are parameterized by the k index, $k = 1, \ldots, k_{max}$, where the elements represent the pre-selected neighborhood structures, and $\mathrm{N}^{\mathrm{k}}(x)$ represents the solutions in kth neighborhood of the current solution x. The idea is to explore the x neighborhood looking for a great location. Given a current solution x, we explore the neighboring solutions belonging to $\mathrm{N}^{\mathrm{k}}(x)$. Whenever a neighboring solution $x' \in \mathrm{N}^{\mathrm{k}}(x)$ rather than x is found, the current solution is updated to x'. If there is no solution improvement within this neighborhood structure N^{k}, the structure change will be made, increasing the index $k \leftarrow k + 1$ [36].

Therefore, in order to improve the solution obtained by the proposed DSA, the VND metaheuristic is applied. The neighbors used in the search are generated by modifying the vector of stockyards and berths. For example, in our research problem, neighborhood structures can be defined as follows:

- K = 1 $\rightarrow$ change of a bit in the vector of the stockyards and berths configuration (see Table 6);
- K = 2 $\rightarrow$ change of two bits in the vector of the stockyards and berths configuration (see Table 6).

After finding a neighbor in a neighborhood structure (new configuration of stockyards and berths), a new planning time is calculated using the DSA (Algorithm 1). The solutions are chosen through two types of local search algorithms, i.e., Best Improvement and First Improvement. In the Best Improvement local search, the algorithm tests all neighboring solutions to arrive at the best solution so that the current solution always changes to the best of all neighboring solutions. A local search is of the First Improvement type, if the current solution change the first improved neighbor solution was found.

Algorithm 2 shows the pseudocode implemented to solve the research problem. It takes as inputs the variables Qn, Qb, Qp, and Qe, with their related demands, types, capacities, and routes, as well as lengths, speeds, and daily capacities.

Algorithm 2: VND(Qn, Qb, Qp, Qe)

```
1  $CPB \leftarrow \mathbb{1}$
2  $H, H_{ls} \leftarrow DSA(Qn, Qb, Qp, Qe, CPB)$
3  $k \leftarrow 1$
4  $k_{max} \leftarrow 2$
5  while $k \leq k_{max}$ do
6      while not the end of the local search do
7          $CPB \leftarrow$ NeighborhoodChange$(CPB, k)$
8          $H' \leftarrow DSA(Qn, Qb, Qp, Qe, CPB)$
9          if $H' < H_{ls}$ then
10             $H_{ls} \leftarrow H'$
11         end if
12     end while
13     if $H_{ls} < H$ then
14         $H \leftarrow H_{ls}$  $k \leftarrow 1$
15     else
16         $k \leftarrow k + 1$
17     end if
18 end while
19 Return H
```

Algorithm 2 begins setting in line 1 the first stockyard and berth configuration vector CPB as a vector of ones, which means that all stockyards available to be used and all types of ships can be moored in any berth (see Table 5). The DSA is called in line 2 for the calculation of the incumbent solution H and the auxiliary local search solution H_{ls}. In lines 3 and 4, we set, respectively, the initial neighborhood $k = 1$ and the use of two neighborhoods to improve the current solution $k_{max} = 2$. The loop from line 5 to 18 runs while a local minimum with respect to all k_{max} neighborhoods is not found. The loop from line 6 to 12 stops when the local search finishes, depending on the applied strategy: Best or First Improvement. In line 7, given the current stockyards and berths configuration CPB and the current neighborhood structure k, function `NeighborhoodChange()` builds a new stockyards and berths configuration CPB not yet used in the current local search loop.

The new solution is calculated in line 8 using Algorithm 1. In line 9, it is verified whether the new solution H' is better than the current local search solution H_{ls}, which is updated in line 10 if an improvement is obtained. When the local search finishes, the algorithm compares the incumbent value H with the new value H_{ls} obtained from the kth neighborhood (line 13). If an improvement is obtained, the incumbent is updated (line 14), and k is returned to its initial value (line 14). Otherwise, the next neighborhood is considered (line 16). The output consists of the best solution found (line 19). Since the function `NeighborhoodChange()` runs, in the worst case ($k = 2$ and Best Improvement), in $O((p+b)^2)$ and uses Algorithm 1 to calculate the objective function, the complete local search is done in $O((p+b)^2) \times [O(np) + O(ne^2) + O(n^2b) + O(n^2) + O(nb)]$ time complexity.

5. Results and Discussion

In all experiments we carried out, the First Improvement local search proved to be more efficient than the Best Improvement local search. Therefore, we analyzed only the results considering the proposed algorithm with the First Improvement local search. All experiments were performed on a personal computer with Windows 10 Pro, Intel® Core™ i7-7700K CPU @ 4.20GHz, 16.0 GB RAM and a 64-bit operating system type with an x64-based processor. The solver used to implement the algorithms was *Pycharm Professional* 2018.3, and the code was programmed in the Python programming language.

In this paper, we consider two different scenarios. The first one has a stockyard–port system configured with 11 stockyards, 3 berths, and 10 h of conveyor belt operation per day. The second scenario addressed represents a stockyard–port system with even more participants. It is configured for 15 stockyards, 5 berths, and the same 10 h of conveyor belt operation per day. For each scenario, a number of ships ranging from 10 to 1000 are considered in the planning period. Each experiment was run 5 times to calculate the mean, standard deviation, and the relative planning time error of the entire planning furnished by the best solutions (relative error of best solutions), i.e., the time spent between the order of the first vessel and the unberthing of the last vessel after algorithm convergence.

5.1. Experiments for the First Scenario (11 Stockyards, 3 Berths, and 10 h of Conveyor Belt Operation per Day)

Table 8 shows, for each ship demand, the best total planning times and their respective averages provided by both initial and best solutions. It also shows the relative errors obtained with the best solutions and the averages of the computational times. From the initial solutions with 10 ships, the shortest total planning time is 58.95 h, and the largest total planning time is 155.28 h. The shortest planning time is due to the better distribution of stockyards and berths, the shorter stacking and reclaiming times, and the greater resources of the conveyor belts. It can be observed in Table 8 that, for 1000 served ships, the *VND* algorithm converged to the average planning time equal to 5338.05 h, with an average computational time of 64.59 s.

Table 8. Total planning time (11 stockyards, 3 berths, and 10 h).

11 Stockyards/3 Berths/10 h/VND						
Ships	**Best Time Initial Solution (h)**	**Best Time Best Solution (h)**	**Average Time Initial Solution (h)**	**Average Time Best Solution (h)**	**Relative Error Best Solution (%)**	**Average Computational Time (s)**
10	58.95	54.97	92.43	65.14	17.63	0.99
50	302.14	260.22	481.30	275.30	4.09	4.19
100	555.49	479.27	1178.46	519.79	8.10	6.29
500	4461.22	2485.64	5863.22	2658.96	5.81	40.28
1000	6819.10	4730.70	10,399.56	5338.05	6.33	64.59

It can be verified in the results shown in Figure 2 that the total planning and computational times provided by the proposed optimization algorithm based on the VND metaheuristic have a linear tendency in relation to the number of ships served.

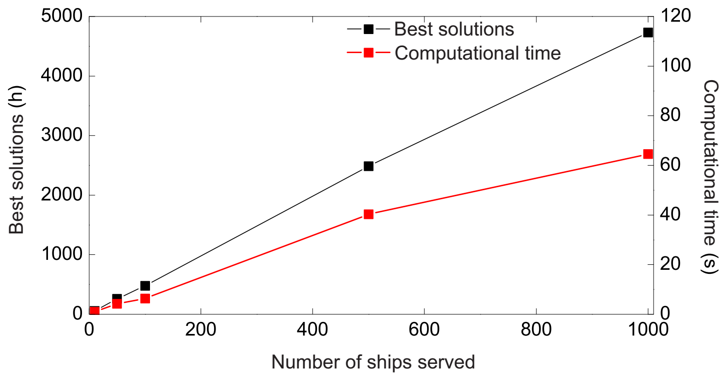

Figure 2. Planning and computational times of best solutions for 11 stockyards and 3 berths.

The average planning time provided by the best solutions decreased by around 50% compared to the initial solutions, except for the case with 10 ships, as shown in Figure 3. For 10 ships, the total average planning time provided by the best solution was reduced by 29.52% compared to the average time provided by the initial solution. For 50, 100, 500, and 1000 vessels, these reductions were 42.8%, 55.89%, 54.65%, and 48.67%, respectively.

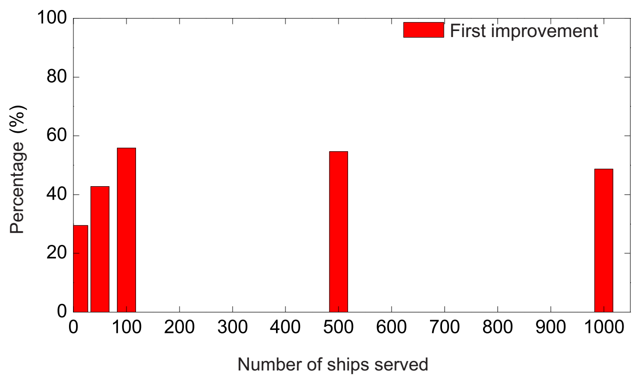

Figure 3. Average percentage of improvement between the initial and best solutions for 11 stockyards and 3 berths versus number of ships served.

Figure 4 shows that the relative error of the planning time provided by the best solutions tends to stabilize as the number of ships increases. It can be observed that the best solutions remain almost constant after removing the solution for 10 ships in the stockyard–port planning. This is because a large number of ships in the berths can be uniformly distributed in the vector of stockpiles and ship types.

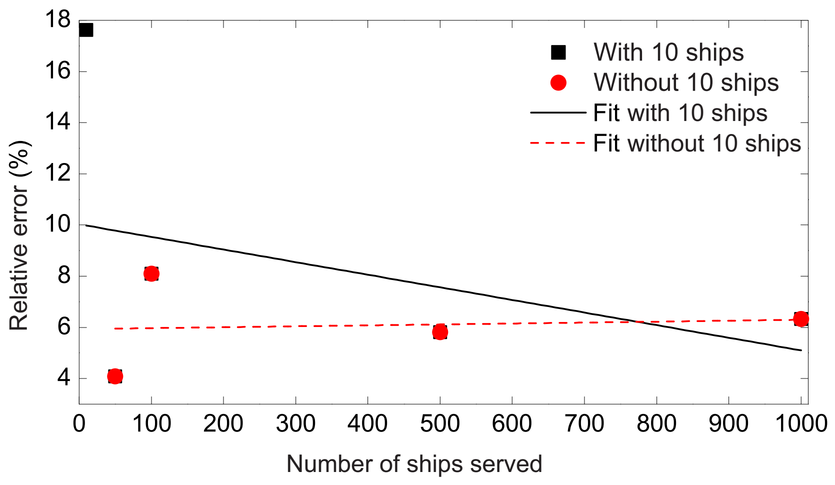

Figure 4. Relative error of planning time versus number of ships served in the first scenario.

5.2. Experiments for the Second Scenario (15 Stockyards, 5 Berths, and 10 h of Conveyor Belt Operation per Day)

Table 9 presents the calculations of the relative errors, showing the same conclusions as Table 8, except for the computational time due to the increase in the stockyards.

Table 9. Total planning time (15 stockyards, 5 berths and 10 h).

15 Stockyards/5 Berths/10 h/VND						
Ships	**Best Time Initial Solution (h)**	**Best Time Best Solution (h)**	**Average Time Initial Solution (h)**	**Average Time Best Solution (h)**	**Relative Error Best Solution (%)**	**Average Computational Time (s)**
10	58.56	53.69	77.6293	60.18	16.61	3.71
50	293.31	195.61	353.88	202.08	4.13	11.57
100	590.532	372.82	772.56	415.79	6.30	22.58
500	3129.749	2158.54	5045.83	2268.02	5.02	103.25
1000	6131.69	3976.82	6822.69	4200.44	4.43	190.14

Figures 5 and 6 compare the planning and computational times, respectively, considering the number of ships served for 11 stockyards and 3 berths, and for 15 stockyards and 5 berths.

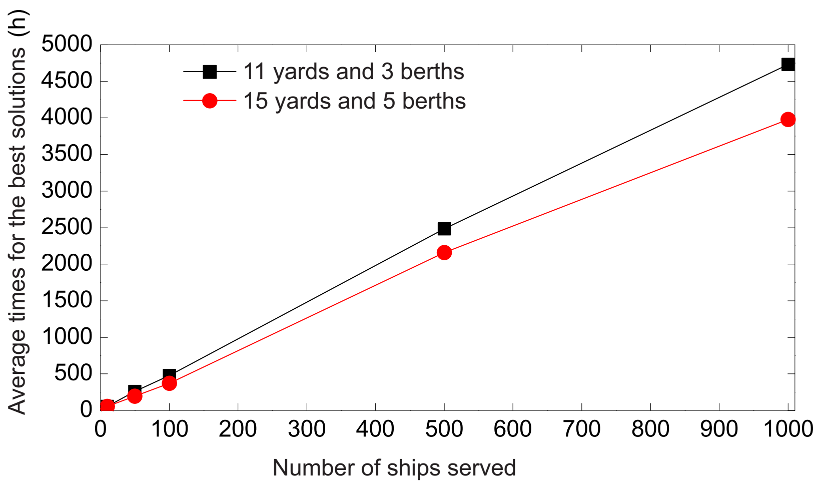

Figure 5. Average planning times of the best solution for all experiments in the first and second scenarios.

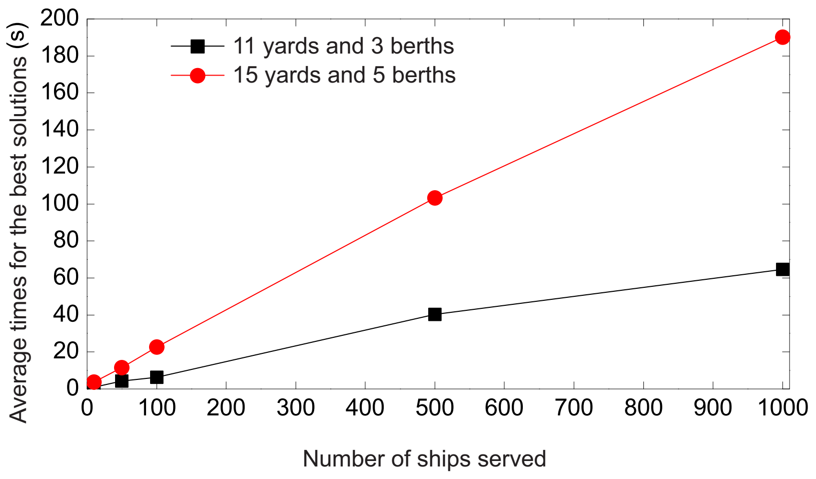

Figure 6. Average computational times of the best solution for all experiments in the scenarios.

Comparing the average planning times provided by the initial and best solutions for this scenario, we also observe a decrease, similar to that observed in the previous case (11 stockyards and 3 berths, see Figure 3). The total average planning time for 10 ships provided by the best solution decreased 22.48% in relation to the average planning time provided by the initial solution. For 50, 100, 500, and 1000 ships, these reductions were 42.89%, 46.17%, 55.05%, and 38.43%, respectively.

5.3. Increasing the Capacity of Conveyor Belts

We also analyzed the effect of increasing the capacity of the conveyor belts according to the planning times obtained through the proposed algorithm. The VND algorithm determined a planning time of 7255.31 h for the case that considers 11 stockyards and 3 berths, with conveyor belts having a capacity of 80,000 tons per day. However, when the conveyor belt's capacity was increased to 160,000 and 320,000 tons, the planning times decreased to 4252.83 h and 2764.83 h, respectively. This represents a decrease of 41.38% and 61.89%, respectively, in comparison to the solution obtained with conveyor belts having a capacity of 80,000 tons. In the case of 15 stockyards and 5 berths, if the conveyor belt's capacity is 80,000 tons per day, the planning time required is 5566.89 h. However, if the conveyor belt's capacity is increased to 160,000 and 320,000 tons, the planning time decreases to 3355.80 h and 2245.63 h, respectively. This represents a decrease of 39.71% and 59.66%, respectively, in comparison to the solution obtained with conveyor belts having a capacity of 80,000 tons.

Figure 7 illustrates the relationship between the daily capacity of the conveyor belt and the initial planning time obtained from the VND optimization algorithm. It is important to note that there is an inverse relationship between the conveyor belt's capacity and the planning time required to handle the same number of ships (1000 ships). Furthermore, an increase in the number of stockyards results in a decrease in the initial planning time, assuming the same ship demand. This is because a larger stockyard requires more conveyor belts and a higher flow of iron ore. Certainly, the metaheuristic should decrease the planning time, but it depends on the stacking and reclaiming times, the capacity of the conveyor belts, and the number of stockyards.

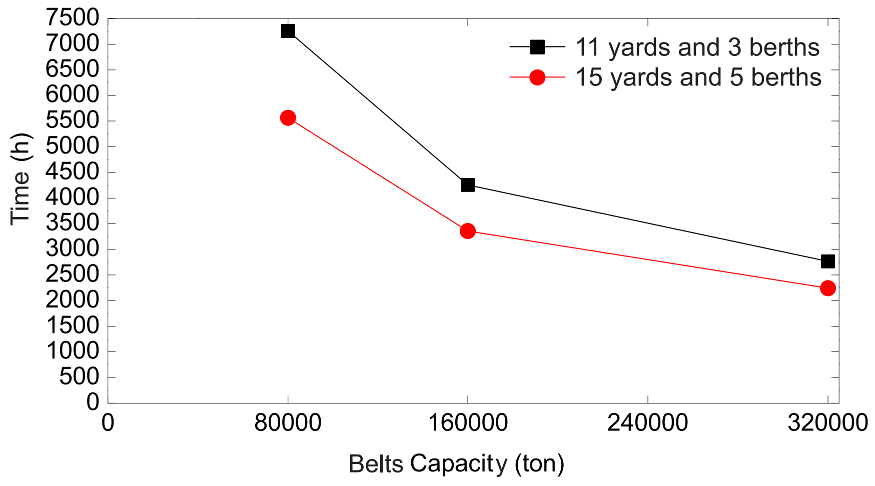

Figure 7. Planning time to attend 1000 ships in the first and second scenarios versus conveyor belt capacity.

From the experiments carried out with a demand of 1000 ships, the solution converged within a short time of around 190 s. This was due to the metaheuristic providing intensification in the solution search and its neighborhoods being very similar. However, there was a lack of diversification because the local First Improvement search was more efficient in obtaining a greater number of better solutions than the Best Improvement search. Therefore, it would be desirable to diversify the neighborhood structures to search for better solutions in other neighborhoods and to create neighbors that are significantly different from each other, thus allowing the algorithm to continue searching for better solutions.

6. Conclusions

In this work, we proposed a computational algorithm to solve the stockyard–port planning problem raised in iron ore exporting companies, considering a long planning horizon in which a large volume of raw material arriving at the yards by trains must be stored and transported to the ships in an efficient and sustainable way. This complex combinatorial problem has been studied primarily using mathematical programming techniques limited by the size of data instances. The proposed algorithm utilizes a deterministic simulator, with the real stockyard–port equipment represented by a graph, and the ships arriving as a first-input-first-output queuing process. The simulator offers great flexibility for decision-makers to quickly analyze different scenarios of the stockyard–port, taking into account different equipment capacities (stackers, reclaimers, and conveyor belts), route diversity, stockyards, and berths. Moreover, it allows for the choice of an initial scenario that meets the amount of iron ore requested by the ships, as well as the maximum transport time of iron ore in the stockyard–port system. Thereafter, the planning time—the time elapsed between the first ship's arrival at the port and the completion of the transshipment operation of the last ship leaving the port—is minimized using the Variable Neighborhood Descent (VND) metaheuristic.

The simulations conducted using the two types of local search revealed that the metaheuristic solutions tended to utilize more stockyards when compared to the initial solution generated by the deterministic simulator. Both local search methods provided an increase in the number of conveyor belts, leading to a higher flow of iron ores to the berths. Furthermore, it was observed that there is an inverse relationship between the capacity of the belts and the planning time required to serve the same number of ships. Therefore, we concluded that increasing the number of conveyor belts is an important strategy to reduce raw material transport time in the stockyard–port system and consequently, to reduce environmental impacts and increase productivity. This positive impact is greater in the case of increasing the number of yards and ship berths.

The convergence time of the VND metaheuristic ranged from 1 to 190 s, with the total number of ships served in the berths varying from 10 to 1000 units, and the number of stockyards and berths varying from 11 to 15 and 3 to 5, respectively. The averages of the total planning times provided by the best solutions obtained by VND increased almost linearly in relation to the number of ships served. For each number of vessels served, the averages of the best solutions decreased by approximately 50% in comparison to the averages of the initial solutions. As the number of ships considered in the planning period increased, the relative error of the best solutions tended to follow a normal distribution.

The proposed algorithm has demonstrated flexibility and speed in solving an NP-hard combinatorial problem, making it suitable for implementation in an iron ore stockyard–port that follows Industry 4.0 principles. It enables real-time decision-making, considering its economic and environmental impacts. The results obtained in this study motivated us to pursue challenges related to the development of a stochastic simulation model, considering uncertainties to improve the evaluated VND algorithm. A performance comparison of different solution techniques in terms of computational effort is also part of our future work, including statistical analysis using simulation software like Arena and/or ANOVA, among others.

Author Contributions: Conceptualization, Á.D.O.L., H.R.O.R. and J.L.F.S.; methodology, Á.D.O.L., H.R.O.R., R.E.N.M. and J.L.F.S.; software, Á.D.O.L. and M.W.J.S.J.; validation, Á.D.O.L., H.R.O.R., M.W.J.S.J., J.A.L.S., R.E.N.M. and J.L.F.S.; formal analysis, Á.D.O.L., H.R.O.R., M.W.J.S.J., J.A.L.S., R.E.N.M. and J.L.F.S.; investigation, Á.D.O.L., H.R.O.R., M.W.J.S.J., J.A.L.S., R.E.N.M. and J.L.F.S.; resources, H.R.O.R. and J.L.F.S.; data curation, Á.D.O.L., H.R.O.R., M.W.J.S.J., J.A.L.S., R.E.N.M. and J.L.F.S.; writing—original draft preparation, Á.D.O.L., H.R.O.R. and J.L.F.S.; writing—review and editing, Á.D.O.L., H.R.O.R., M.W.J.S.J., J.A.L.S., R.E.N.M. and J.L.F.S.; visualization, H.R.O.R., J.A.L.S., R.E.N.M. and J.L.F.S.; supervision, H.R.O.R. and J.L.F.S.; project administration, H.R.O.R. and J.L.F.S.; funding acquisition, H.R.O.R. and J.L.F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by FAPES-2022-BWBR2, FAPES-2021-WMR44, CNPq 309737/2021-4 and 309490/2021-9 projects, and also by FAPES and Vale Company in Brazil, grant number 529/2016.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors acknowledge the support from LabTel-UFES, NiDA, and Research Centre Biobased Economy from the Hanze University of Applied Sciences.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Issaoui, Y.; Khiat, A.; Bahnasse, A.; Ouajji, H. Toward smart logistics: Engineering insights and emerging trends. *Arch. Comput. Methods Eng.* **2021**, *28*, 3183–3210. [CrossRef]
2. Rüßmann, M.; Lorenz, M.; Gerbert, P.; Waldner, M.; Justus, J.; Engel, P.; Harnisch, M. Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston Consult. Group* **2015**, *9*, 54–89.
3. Marinagi, C.; Reklitis, P.; Trivellas, P.; Sakas, D. The Impact of Industry 4.0 Technologies on Key Performance Indicators for a Resilient Supply Chain 4.0. *Sustainability* **2023**, *15*, 5185. [CrossRef]
4. Khan, Y.; Su'ud, M.B.M.; Alam, M.M.; Ahmad, S.F.; Ahmad Ayassrah, A.Y.A.B.; Khan, N. Application of Internet of Things (IoT) in Sustainable Supply Chain Management. *Sustainability* **2023**, *15*, 694. [CrossRef]
5. Filipov, V.; Vasilev, P. Manufacturing operations management—The smart backbone of Industry 4.0. *Int. Sci. Conf. Ind. 4.0* **2016**, *1*, 71–76.
6. De Paula Ferreira, W.; Armellini, F.; De Santa-Eulalia, L.A. Simulation in industry 4.0: A state-of-the-art review. *Comput. Ind. Eng.* **2020**, *149*, 106868. [CrossRef]
7. Nota, G.; Nota, F.D.; Peluso, D.; Toro Lazo, A. Energy efficiency in Industry 4.0: The case of batch production processes. *Sustainability* **2020**, *12*, 6631. [CrossRef]
8. Csalódi, R.; Süle, Z.; Jaskó, S.; Holczinger, T.; Abonyi, J. Industry 4.0-driven development of optimization algorithms: A systematic overview. *Complexity* **2021**, *2021*, 6621235. [CrossRef]
9. Xu, J.; Huang, E.; Hsieh, L.; Lee, L.H.S.; Chen, C.H. Simulation optimization in the era of Industrial 4.0 and the Industrial Internet. *J. Simul.* **2020**, *10*, 310–320. [CrossRef]
10. Iris, Ç.; Lam, J.S.L. A review of energy efficiency in ports: Operational strategies, technologies and energy management systems. *Renew. Sustain. Energy Rev.* **2020**, *112*, 170–182. [CrossRef]
11. Leal Gomes Leite, J.M.; Arruda, E.F.; Bahiense, L.; Marujo, L.G. Modeling the integrated mine-to-client supply chain: A survey. *Int. J. Mining Reclam. Environ.* **2020**, *34*, 247–293. [CrossRef]
12. Koch, B.; Mehendiratta, V. Solving the Sotckyard Optimization Trilemma. Mining Magazine. Available online: https://www.miningmagazine.com/tag/stockyard (accessed on 23 January 2023).
13. Ago, M.; Nishi, T.; Konishi, M. Simultaneous optimization of storage allocation and routing problems for belt-conveyor transportation. *J. Adv. Mech. Des. Syst. Manuf.* **2007**, *1*, 250–261. [CrossRef]
14. Servare, M.W.J., Jr.; Rocha, H.R.D.O.; Salles, J.L.F.; Perron, S. A Linear Relaxation-Based Heuristic for Iron Ore Stockyard Energy Planning. *Energies* **2020**, *13*, 5232. [CrossRef]
15. Boland, N.; Gulczynski, D.; Savelsbergh, M. A stockyard planning problem. *EURO J. Transp. Logist.* **2012**, *1*, 197–236. [CrossRef]
16. Boland, N.; Gulczynski, D.; Jackson, M.P.; Savelsbergh, M.W.P.; Tam, M.K. Improved stockyard management strategies for coal export terminals at Newcastle. In Proceedings of the 19th International Congress on Modelling and Simulation, Perth, Australia, 12–16 December 2011; pp. 718–724.
17. Hanoun, S.; Khan, B.; Johnstone, M.; Nahavandi, S.; Creighton, D. An effective heuristic for stockyard planning and machinery scheduling at a coal handling facility. In Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN), Bochum, Germany, 29–31 July 2013; pp. 206–211.
18. Belov, G.; Boland, N.; Savelsbergh, M.W.; Stuckey, P.J. Local search for a cargo assembly planning problem. In Proceedings of the 11th International Conference, CPAIOR, Cork, Ireland, 19–23 May 2014; pp. 159–175.
19. Belov, G.; Boland, N.; Savelsbergh, M.W.P.; Stuckey, P.J. Exploration of models for a cargo assembly planning problem. *arXiv* **2015**, arXiv:1504.00445.
20. Savelsbergh, M.; Smith, O. Cargo assembly planning. *EURO J. Transp. Logist.* **2015**, *4*, 321–354. [CrossRef]
21. Menezes, G.C.; Mateus, G.R.; Ravetti, M.G. A branch and price algorithm to solve the integrated production planning and scheduling in bulk ports. *Eur. J. Oper. Res.* **2017**, *258*, 926–937. [CrossRef]
22. Moreno, E.; Rezakhah, M.; Newman, A.; Ferreira, F. Linear models for stockpiling in open-pit mine production scheduling problems. *Eur. J. Oper. Res.* **2017**, *260*, 212–221. [CrossRef]
23. Le, V.T.; Johnstone, M.; Zhang, J.; Khan, B.; Creighton, D.; Hanoun, S.; Nahavandi, S. Complex simulation of stockyard mining operations. In *Advances in Global Optimization*; Springer: Cham, Switzerland, 2015; pp. 529–537.
24. Bai, X.; Zhao, Y.; Liu, Y. A novel approach to study real-time dynamic optimization analysis and simulation of complex mine logistics transportation hybrid system with belt and surge links. *Discret. Dyn. Nat. Soc.* **2015**, *2015*, 601578. [CrossRef]
25. Van Vianen, T.; Ottjes, J.; Lodewijks, G. Belt conveyor network design using simulation. *J. Simul.* **2016**, *10*, 157–165. [CrossRef]
26. Balzary, J.; Mohais, A. Consideration for multi-objective metaheuristic optimisation of large iron ore and coal supply chains, from resource to market. In *Advances in Applied Strategic Mine Planning*; Springer: Cham, Switzerland, 2018; pp. 297–316.
27. Gao, Z.; Sun, D.; Zhao, R.; Dong, Y. Ship-unloading scheduling optimization for a steel plant. *Inf. Sci.* **2021**, *544*, 214–226. [CrossRef]
28. Sun, D.; Meng, Y.; Tang, L.; Liu, J.; Huang, B.; Yang, J. Storage space allocation problem at inland bulk material stockyard. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *134*, 101856. [CrossRef]
29. Silva, A.; Beneteli, T.A.P.; Silva, L.; Pessin, G.; Euzébio, T.A.M.; Cota, L.P. A mixed-integer linear programming model for the stockpiles blending problem in a nickel mine. *Int. J. Min. Miner. Eng.* **2022**, *13*, 93–118. [CrossRef]
30. Tang, X.; Jin, J.G.; Shi, X. Stockyard storage space allocation in large iron ore terminals. *Comput. Ind. Eng.* **2022**, *164*, 107911. [CrossRef]

31. Hansen, P.; Mladenović, N.; Brimberg, J.; Pérez, J.A.M. Variable neighborhood search. *Comput. Oper. Res.* **2019**, *24*, 57–97.
32. Holmes, R.J.; Lu, Y.; Lu, L. Introduction: Overview of the global iron ore industry. In *Iron Ore*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 1–56.
33. Santos, M.S.; Pinto, T.V.; Júnior, Ê.L.; Cota, L.P.; Souza, M.J.; Euzébio, T.A. Simheuristic-based decision support system for efficiency improvement of an iron ore crusher circuit. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103789. [CrossRef]
34. Pereira, F.G.G.; Cruz, J.P.G.; Botter, R.C.; Robles, L.T. Optimization model for integrated port terminal management. *Trends Marit. Technol. Eng.* **2022**, *2*, 75–83.
35. Guze, S. Graph theory approach to the vulnerability of transportation networks. *Algorithms* **2019**, *12*, 270. [CrossRef]
36. Gendreau, M.; Potvin, J.Y. *Handbook of Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 2.