Optimising discrete dynamic berth allocations in seaports using a Levy Flight based meta-heuristic

Ran Wang[a,1], Trung Thanh Nguyen[a,1,*], Changhe Li[b], Ian Jenkinson[a], Zaili Yang[a], Shayan Kavakeb[c]

[a] *Liverpool Logistics Offshore and Marine Research Institute (LOOM), School of Engineering, Technology and Maritime Operations, Liverpool John Moores University, L3 3AF, United Kingdom*
[b] *School of Computer Science, China University of Geosciences, Wuhan, China*
[c] *AECOM, United Kingdom*

Abstract

Seaports play a vital role in our everyday life: they handle 90% of our world trade goods. Improving seaports' efficiency means improving the efficiency of sending and receiving our goods. In seaports, one of the most important and most expensive operations is how to allocate vessels to berths. In this paper, we solve this problem by proposing a new meta-heuristic, which combines the nature-inspired Levy Flight random walk with local search, while taking into account tidal windows. With our algorithm, we meet the following goals: (i) to minimise the cost of all vessels while staying in the port, and (ii) to schedule available berths for the arriving vessels taking into account a multi-tidal planning horizon. In comparison with the state-of-the-art exact method using commercial solver and a competitive heuristic, the computational results prove our approach guarantees feasibility of solutions for all the problem instances and is able to find good solutions in a short amount of time, especially for large-scale instances. We also compare our results to an existing state-of-the-art Particle Swarm Optimisation and our work produces significantly better performances on all the test instances.

Keywords: Levy flight, Berth allocation problem, Tidal windows, Meta-heuristic algorithms, Dynamic Optimisation

1. Introduction

Nowadays seaports handle over 90% of world's trade (UNCTAD, 2015) and they keep expanding the scale of import and export, hence, optimising the efficiency of port operations is vital for the mobility of goods. One of the most important operations in seaports is the berth allocation problem (BAP), which is to schedule vessels to suitable berths. On the port side, an efficient schedule determines how many vessels can be accommodated in one day and how much cost or benefit it will produce. On the fleet side, the schedule determines the time the vessel should arrive the berth, which berth to arrive and the associated cost.

The berth allocation can be the most expensive one out of all port operations, because if a vessel cannot be admitted in time, the port will have to pay heavy penalty. If a vessel has to stay for long (e.g. due to low tides or congestions due to other incoming vessels), the shipping schedule can be delayed and an entire good supply chain can be affected internationally. In this case minimising the cost is very significant.

To better understand the problem, we explain the procedure of the berth allocation operation and then introduce the variants of BAPs. The aim of BAPs is to schedule a set of vessels that are arriving to a terminal of a port. Before arrival, vessels will notify the terminal of the estimated arrival time and other necessary information to be able to assign a

*Corresponding author
Email addresses: r.wang@2015.ljmu.ac.uk (Ran Wang), t.t.nguyen@ljmu.ac.uk (Trung Thanh Nguyen), Changhe.lw@gmail.com (Changhe Li), i.d.jenkinson@ljmu.ac.uk (Ian Jenkinson), z.yang@ljmu.ac.uk (Zaili Yang), shayan.kavakeb@aecom.com (Shayan Kavakeb)
[1] Co-first authors. These two authors contributed equally to this work.

valid berth as soon as possible. While berthing, the handling time of a vessel, which is the time needed for cargo loading and unloading, can be fixed or variable depending on the resource arrangement. For example, in order to load or unload goods, a vessel may need to wait because resources such as certain type of quay crane may not be available yet. In addition, the water depth sometimes affects the availability of a berth because each vessel has a different draught and hence it can only stay in a berth with a deep enough water level. The water level at a berth can vary because of changing tides. In other words, the tides determine the availability of berths for each vessel. Therefore, considering tidal constraints is essential to allocate incoming vessels to feasible berthing positions.

Variants of BAPs have been categorised based on criteria such as discrete or continuous, static or dynamic, deterministic or stochastic [1, 2]. Discrete BAPs separate a quay into a number of berths with certain lengths so that vessels are able to moor at one of the berths. A vessel is not able to occupy more than one berth and a berth can only serve one vessel at a time. In continuous BAPs, the quay is treated as a whole with a certain length. Based on the length of each vessel, vessels can berth at arbitrary positions. Hybrid BAPs are similar to discrete BAPs in that the quay is separated into berths. In hybrid BAPs, two adjacent berths are allowed to combine to serve a vessel if the vessel is too long to stay at a single berth. In terms of the arrival time of vessels, the BAP is also categorised into static and dynamic [3]. Static BAPs generally assume that all the vessels have been arrived to the port from the beginning of the time horizon. On the other hand, in dynamic BAPs vessels arrive as time goes by. It means that a vessel cannot berth before its arrival time. A variety of constraints reflecting the real-world problem were also summarised in [1, 2].

Our work in this paper will mainly focus on discrete dynamic BAPs taking into account tidal constraints. Among existing approaches, the latest approximate algorithm [4] has shown decent results on BAPs with two-tide constraints. Because it only accommodates two tides, it sometimes obtains infeasible solutions when the number of vessels increases. The approach introduced in [4] is a greedy heuristic which allocates vessels in a predefined order (processing time of the vessel / unit cost of the vessel). For each vessel, the heuristic chooses an available berth with the minimum increment of the objective value. The final schedule is totally determined by the fixed order of adding vessels without any stochastic element. Unlike meta-heuristics, there is no other operation used in [4] to improve the only solution. In this paper our algorithm will deal with multiple tidal windows whereas [4] only deals with two tides. The new meta-heuristic also brings more randomness to diverse the solution while [4] is deterministic. Another state-of-the-art exact method [5] using commercial solver ensures feasible solutions but the capability of dealing with large-scale problems is limited and the computation time is too long to obtain good solutions. In this paper, we propose a new meta-heuristic to improve the limitations of existing methods. We propose a new approximate approach that both diversifies an existing deterministic heuristic and helps that heuristic converge more efficiently. It is a single-solution based meta-heuristic which combines Levy flight random walk with local search. We believe that this is the first work of combining heuristic in [4] with Levy flight and a new local search. Due to the nature of the deterministic heuristic, it is also innovative that we encode the solution as priorities of assigning vessels.

The main contributions of our algorithm are summarised as follows. Firstly our algorithm provides competitive berth allocation schedules comparing to the state-of-the-art exact and approximate methods. Secondly it is the best algorithm so far that can always achieves feasible solutions for both small-scale and large-scale problems in a short running time. Furthermore, it is also the only algorithm that is able to provide good quality solutions for the large-scale cases.

The structure of this paper is as follows. A study of related existing work is shown in Section 2. In Section 3 and Section 4, we will describe our problem and our approach in detail. The experiment is carried out in Section 5 including comparisons with other works and analysis. Section 6 highlights our contribution and proposes potential future works.

2. Literature review

BAPs draw a lot of attentions from the academic community, evidenced by the large amount of existing work on solving this problem. Both the discrete and continuous BAPs have been proved to be NP-hard [6, 7]. The survey paper [1, 2] already included the majority of work relating to BAPs. So in this section we only briefly review recent developments

in solving BAPs, followed by a discussion on relevant work that dealt with the problem considered in this paper. The static BAP (SBAP) was formulated in [8] and it was extended as a dynamic BAP (DBAP) firstly in [3]. An improvement of [3] was made in [9]. A Lagrangian relaxation technique is applied to the mathematical formulation and a genetic algorithm (GA) is proposed to solve the problem.

In order to deal with the limited availability of berths, we focus on reviewing relevant work. A BAP with dynamic arrival time and due dates was addressed in [10]. It presented a tabu search method and was tested on a data set from a terminal in Port of Gioia Tauro, Italy. The tabu search is improved by adding an elite set of solutions to the path relinking algorithm and a swap move to the local search [11]. A variable neighbourhood search heuristic was proposed by [12] aiming to minimise the cost with the constraint which berths start to be available from different time. [13] developed a GA based heuristic while considering the berth availability and the priority of vessel services. [14] presented a clustering search to solve a BAP with time windows. The clustering search consists of a simulated annealing to generate solutions, a clustering process and a local search. [15] proposed a particle swarm optimisation (PSO) to solve a BAP. It was tested on the data set from [10] with an objective of minimising the total service time. [16] applied a GA to a tactical BAP aiming to determine berth positions, berth time and allocations of quay cranes for incoming vessels. The tactical BAP aims to allocate vessels to their favourite berth positions as vessels are expected to arrive periodically. [17] formulated the tactical BAP while considering uncertain dwell time of vessels. Another BAP considering uncertain vessels' arrival time and operation time was solved by [18] with a two-stage decision model and a meta-heuristic approach for large-scale problems. [19] proposed a robust algorithm integrating a simulated annealing algorithm and a branch-and-bound algorithm.

Instead of limited available time of berth, tidal window is another way of representing the availability of berths. In BAPs with tidal windows, each berth offers different availability to vessels periodically based on the tides. This is a more realistic way of modelling the problem. However, there are very few publications actually taking the tides into consideration. [4] applied a greedy heuristic to minimise the total cost of waiting time of all vessels. It considered only two tidal windows (one low tide and one high tide) but the time horizon of the second tide goes to infinity. [20] pointed out the limitation in [4] and presented a POPMUSIC (partial optimisation meta-heuristics under special intensification conditions) framework while limiting the time horizon to two times of the tidal window. [21] also applied a POPMUSIC method to the same problem as in [4]. The POPMUSIC framework was firstly proposed by [22] and it starts from a random permutation and to be improved by solving a mathematical formulation. A generalised set-partitioning approach was proposed in [5] to solve the problem with multiple tides using the commercial solver CPLEX. Apart from tidal windows, [5] also worked on vessel time windows where serving each vessel has to be finished by the due date. There is an earlier work on time windows [23] which was motivated by a real problem at the maritime industrial port complex of Sao Luis. A simulated annealing-based heuristic was used. A more recent dynamic programming-based meta-heuristic was developed in [24] which is capable of dealing with up to 150 ships and 15 berths. [25] proposed an Integer Programming and Constraint Programming for BAPs with changing water depths over tides. Moreover, a machine learning-based system was also applied in a bulk BAP in [26]. The system is trained by running a set of available algorithms and then it provides its best solution for each problem. The brief literature review above shows that meta-heuristics are the preferred methods to solve the BAP in a majority of existing research. This agrees with the finding from [2] which stated the reason for the dominant use of meta-heuristics is their ability to deal with large scale problems within a relatively short amount of time.

Sometimes according to terminal operation demand, a small processing time is necessary, which makes meta-heuristics a beneficial way of solving the problem. To the best of our knowledge, there is no existing work using meta-heuristics to tackle the BAP with multiple tides that can provide feasible solutions for all the test cases. Most work using exact methods are not able to solve large scale data set or reach the optimal solution in a reasonable time because of the limitation of exact techniques.

3. Problem description

In our work, we model the berth allocation with multiple tides as a discrete problem. The problem is described by introducing assumptions, notations and the mathematical formulation of objective function as follows.

3.1. Assumptions

1. One berth can only serve one vessel at a time.
2. The processing time of a vessel is the same no matter which berth it goes to.
3. Once a vessel has started the serving process, it cannot be interrupted.
4. Berths will become available right after a vessel has been served.
5. All berths are available from the initial time 0.
6. The time horizon is from 0 to infinity until all vessels are scheduled.
7. There is at least one berth available for each vessel at low tides

3.2. Notations

m:	The total number of berths
n:	The total number of vessels
t_arr_j:	The arrival time of vessel j
t_wait_j:	Waiting time of vessel j which equals to t_start_j-t_arr_j
t_proc_j:	Processing time of vessel j
w_j:	Unit cost of vessel j. A time-scaled cost is generated after vessel j arrives at the terminal
TF:	Tide changing frequency
L_j:	The indicator of the availability of vessel j at all berths at low tide
H_j:	The indicator of the availability of vessel j at all berths at high tide

The decision variables are shown as below:

t_start_j:	Start time to serve vessel j
x_{ij}:	Equals to 1 if vessel j is assigned to berth i and 0 otherwise
$I_{ijj'}$:	Equals to 1 if vessel j and $j^{'}$ are both assigned to berth i and vessel j is processed before vessel $j^{'}$, and 0 otherwise

3.3. Mathematical model

Due to the strong impact of changing tides in practice, we consider it in our problem as a restriction to vessels. In our problem setting, high tides and low tides happen alternatively. According to Section 3.2, TF denotes tide changing frequency. In this way, the time horizon will be divided into $[0, TF)$, $[TF, 2 * TF)$ ... until all vessels have been scheduled. For example, if $TF = 12$, the water level alternates between low and high in every 12 hours. Because of the variety of draught of vessels, not all the berths are available for a vessel at all time. It means that at low tide and high tide, there are certain berths that can be available to only a specific type of vessel. Suppose a set of m berths have been sorted in ascending order of the water depth and denoted as 1, 2, 3, 4 ... m (m is integer). There are a set of n vessels. Let L_j be the indicator of the availability of vessel j at all berths at low tide ($1 \leq j \leq n$, j is integer), then the set of berths that vessel j can visit at low tide is defined as $SL_j = \{L_j, L_j + 1, ..., m\}$. For high tide we define H_j as the indicator of vessel j so that the set of berths can be summarised as $SH_j = \{H_j, H_j + 1, ..., m\}$. For example, as show in Table 1, when vessel ID $j = 5$, $L_5 = 4$ indicates that at low tide only berth 4 is available for this vessel, while at high tide $H_5 = 2$ indicates that berth 2, 3, 4 are available. Note that we assume that the water level of a berth at high tide is always higher than that at low tide. It means that for a vessel the amount of berths available at high tide must be no less than at low tide.

The objective function (1) minimises the total cost of the service time of each vessel from the time it arrives until the time it finishes all the loading and unloading. The service time includes the waiting time and processing time.

$$\min \sum_{j=1}^{n} w_j(t\_wait_j + t\_proc_j) \tag{1}$$

s.t.

$$\sum_{i}^{m} x_{ij} = 1 \quad \forall j \in n \tag{2}$$

$$t\_start_j \geq t\_arr_j \quad \forall j \in n \tag{3}$$

$$t\_start_{j'} \geq t\_start_j + t\_proc_j - m(1 - I_{ijj'}) \quad \forall j, j' \in n, j \neq j', \forall i \in m \tag{4}$$

$$I_{ijj'} + I_{ij'j} \leq \frac{1}{2}(x_{ij} + x_{ij'}) \quad \forall j, j' \in n, j < j', \forall i \in m \tag{5}$$

$$I_{ijj'} + I_{ij'j} \geq x_{ij} + x_{ij'} - 1 \quad \forall j, j' \in n, j < j', \forall i \in m \tag{6}$$

$$x_{ij} = 0 \quad \forall j \in n, i = 1, 2, ..., max(L_j, H_j) - 1 \tag{7}$$

$$x_{ij} \in \{0, 1\} \quad \forall j \in n, \forall i \in m \tag{8}$$

$$I_{ijj'} \in \{0, 1\} \quad \forall j, j' \in n, \text{s.t.} j \neq j', \forall i \in m \tag{9}$$

Constraint (2) ensures each vessel is assigned to only one berth. Constraint (3) requires that the vessel can only be served after it arrives. Constraint (4) guarantees that if vessel j and $j^{'}$ are assigned to the same berth and vessel j is served before $j^{'}$, then the starting time of serving vessel $j^{'}$ cannot be no earlier than $t\_start_j + t\_proc_j$. Constraint (5) and (6) ensure that one of $I_{ijj'}$ and $I_{ij'j}$ equals to 1 if vessel j and j' are both assigned to berth i. They also ensure that $I_{ijj'} = I_{ij'j} = 0$ if vessel j or vessel $j^{'}$ is not assigned to berth i. Constraint (7) restricts vessel j to be only assigned to a berth always available to it. Constraint (8) and (9) enforce x_{ij} and $I_{ijj'}$ to be binary.

Table 1: Example of available berths to vessels

Vessel ID	j	1	2	3	4	5
Berth indicator at low tide	L_j	1	4	2	4	4
Berth indicator at high tide	H_j	1	1	1	3	2

3.4. The sensitivity of tidal constraints to BAPs

Due to the changing tides, the feasible intervals and forbidden intervals for each variable are intertwined. It makes finding good solutions in the search space very difficult. Especially when the number of vessels increases the computational complexity is usually high. With the goal to minimise the cost (1), the objective value depends on the total weighted start time of all vessels $\sum_{j=1}^{n} w_j * t\_start_j$ since $t\_wait_j = t\_start_j - t\_arr_j$, and the arrival time t_arr_j and process time t_proc_j are constants. The tidal constraints not only make some berths unavailable for some vessels, but also cause changes on the start time of multiple vessels. Assume a vessel is not able to stay at a berth at low tide, it has to wait till high tide and other vessels scheduled after this vessel have to wait too. It obviously increases the total cost. In other words, the total cost is highly sensitive to the changes of start time of vessels.

Algorithm 1 The process of LF-BAP

```
current_solution := Initialisation(); //encoding (Section 4.1.1)
Let best_solution be the best solution so far
Let best_cost be the objective value of best_solution
While (stopping_criteria_not_met) do:
  new_solution = LF_random_walk(current_solution); //Algorithm 2
  dec_solution = Decoding(new_solution); //Algorithm 3
  Local_search(dec_solution); //Algorithm 4
  Evaluate the objective value of dec_solution and denote it by new_cost
  If (new_cost < best_cost) do:
    //update best known solution
    best_cost = new_cost;
    best_solution = new_solution;
  End
  current_solution = new_solution;
End
```

Algorithm 2 The pseudocode of *LF_random_walk()*

```
N := step_length //calculated using (11)
For (int i = 0; i < N, i++ ) do:
  randomly pick two positions in current solution
  switch the contents of them
End
```

4. Meta-heuristic for BAP

We propose a single-solution based meta-heuristic optimisation termed LF-BAP which is based on a random walk named Levy flight. This random walk is based on a long tail distribution which can be used to help an algorithm to escape from getting stuck at a local optimum [27]. The frequency and the length of long jumps are controlled by adjusting the parameters of the distribution. The Levy flight optimisation process describes a move strategy in which a particle flies from one point to another following LF distribution. In the process of LF-BAP only one individual is used. According to our brief review in Section 2, population-based meta-heuristic algorithms have been applied to BAPs by many researchers. In BAPs, however, even a small move of an individual, which is equal to a small change to the start time of a vessel, may cause a major change to the berth allocation plan. This can make population-based methods slow to

converge. Due to the above reason, many existing population-based methods on BAPs are time-consuming. Being a single-solution search, LF-BAP has the potential to avoid the aforementioned problem of slow-convergence on this particular problem.

The pseudocode of the proposed LF-BAP is summarised in Algorithm 1. The process of LF-BAP is terminated when certain criteria is met. Here the algorithm will stop running if one of the following criteria is met: 1) the limit of iteration number is reached; 2) our approach has found the global optimum (provided by the exact technique from the commercial solver, if it is able to solve the problem); 3) the best solution has not been improved for a quarter of the maximum number of iterations.

Each generation of LF-BAP consists of two phases. Due to the large search space caused by the tidal constraints, the first phase aims to efficiently explore good regions in the search space and maintain the diversity of scheduling vessels. It starts from encoding an initial solution (Section 4.1.1) and then updates the current solution by adapting a Levy flight random walk (Section 4.1.2). Our decode procedure (Section 4.1.3) always ensures a feasible solution. To intensify the current solution, three local search procedures are applied in the second phase of our algorithm. A deep exploration in the local search space further improves the schedule.

4.1. First phase

4.1.1. Encoding

To be able to solve the BAP using a meta-heuristic like LF-BAP, we need to find a method to encode all information of a solution into a data structure. In this paper, we propose the following encoding structure: information is stored in an array X. The indices of X indicate the IDs of the vessels, while the value of X[j] indicates the priority of vessel j. The priorities determine the order that the vessels should be allocated to berths. The smaller the value is, the higher the priority is given to the vessel. An example of this encoding is shown in Table 2. In the example, the order of allocating vessels should be vessel 3, 1, 4, 2 and 5 based on their priority. In the initial solution, the priority of vessels are ordered by their arrival time. The vessel with the earliest arrival time has the highest priority in the order.

Table 2: Example of encoding a solution given priorities of vessels

Vessel j	1	2	3	4	5
Priority	2	4	1	3	5
↓↓					
Vessel allocation order	3	1	4	2	5

4.1.2. Adapting Levy flight walks to the BAP

To update the encoded solution, LF-BAP will undertake a random walk of Levy flight by calling the function *LF_random_walk()*. The pseudocode is shown in Algorithm 2. Levy flight is one of the keys in our algorithm to maintain the randomness and the diversity of the berth allocation optimisation.

Levy flight is a concept of a random walk under a certain probability distribution. It is especially useful for natural phenomena and artificial facts such as earthquake analysis, financial mathematics, signal analysis, fluid dynamics, etc. Levy flight search strategy contributes a lot to food pathing in nature-inspired algorithms like Artificial bee colony algorithm [28] and Cuckoo search algorithm [29]. Moreover, Levy flight makes an improvement in the field of computer science. For example, Levy flight was applied in examining the variability of internet traffic [30]. Levy flight was combined with artificial potential field method in order to perform an efficient searching algorithm for multi-robot applications [31]. As mentioned above, lately Levy flight has been effectively applied to optimisation problems with large search space as it significantly increases the diversity of the chromosome and avoids to trap in local optima [32, 33]. However, to the best of our knowledge there are not many studies using Levy flight in global optimisation as single-solution based meta-heuristics except for the research set out in [27]. The distribution below (10) from [27] is a normalisation of the random walking length distributed in [34]:

$$P(l) = \frac{\beta}{l_0(1+\frac{l}{l_0})^{1+\beta}} \tag{10}$$

where l is noted as the step length.

Moreover, [27] also provided a formulation (11) to generate l where U is a standard uniform distribution. l_0 is introduced as a scale factor. The range of step length is $[0, +\infty)$. The probability of achieving a long step decreases when β increases.

$$l = l_0(\frac{1}{U^{1/\beta}} - 1) \tag{11}$$

In general, Levy flight technique is designed for continuous optimisation problems. To apply LF-BAP to the combinatorial domain of the BAP, we round a randomly generated step length to an integer number. This number represents the number of swaps of two random positions in vessel allocation order (Table 2). This ensures that all step lengths are discrete values. The larger the step length, the greater the number of swaps. The more swaps to be made to a solution, the more different the new solution will likely be in comparison to the original solution.

4.1.3. Decoding

We also need to decode the information from the data structure to a berth allocation solution. For this purpose, we modify the representation in [4]. The idea of allocating a vessel to an available berth with the minimum increment of the objective value in [4] is used in our decoding process. However, there are only two tides considered in [4]. All the vessels not allocated in the first tide will be scheduled in the second tide. This way of allocation has a major drawback: solutions can be infeasible under certain circumstances as explained in [5]. There are only two tides in [4] in which the second tide goes to infinity. If there is a large amount of vessels arriving, and the first tide can only accommodate a small portion of them, the second tide has to be much longer than usual. It is not suitable for real-world problems.

As we have more than two tides in reality, to avoid infeasible solutions, we include additional checks that need to be done every time the tide changes. Every tide is restricted to the same length based on the tide changing frequency. When the tide changes from high to low, we have to check whether the finish time of vessels exceeds the current tide because the same berth may become unavailable for the vessel. Furthermore, for both situations (low tide changing to high tide and vice versa) we check whether there is any vessel with a start time exceeding the current tide. They will be removed from the current schedule and be allocated in the next iteration. The pseudocode of our proposed decoding procedure is described in Algorithm 3.

4.2. Second phase

To further improve the schedule, the second phase is applied to the decoded solution. It seeks to make most of the time horizon and find the local optima. When changing from one tide to another, vessels exceeding the current tide will be removed because the same berth at the upcoming tide may not be available. [4] proposed in the last step of their algorithm to remove idle time without violating any constraints by shifting vessels to an earlier time. However, for consecutive vessels it is unlikely that the start time of all of the following vessels is allowed to move up due to the tidal constraints. For instance, there is a vessel that fits in an earlier idle time period but there are other vessels in between. If the vessels in between are not able to move, the solution cannot be improved following the steps in [4].

We propose a new local search which consists of three parts: swap in berth, swap between two berths and move vessels from one berth to another (Algorithm 4). The first two parts were proposed in [15] and the third part is newly proposed in this paper. A swap in berth allows a sequence of vessels assigned to the same berth to swap every two positions. Among all feasible solutions, the pair of vessels with the best improvement in terms of the objective value is swapped. It does the same for each berth in the schedule. Regarding swapping between berths, for every combination of two berths, two vessels are selected randomly and swapped. The swap is kept only if there is an improvement. This procedure is repeated 20 times in the following experiment.

The third part of the local search is to move vessels from one berth to another. We randomly choose a vessel from a certain berth and insert it to a random position in the schedule of another berth if it provides a feasible and better result. The strategy of selecting a berth to be inserted is to pick a berth with a lower water level which is usually less busy. For each group of an original berth and a lower-water-level berth to be inserted to, we pick one random vessel from the original berth and one random position from the lower-water-level berth. For example, assume that there are two berths b_1 and b_2 with a lower water level than berth b_3. We randomly choose a vessel currently allocated to b_3 and insert it to a random position in b_2. Similarly for b_1, we randomly choose a vessel from b_3 and try to insert it to b_1. If a move like this leads to an

Algorithm 3 The pseudocode of the decoding procedure

Let V be the solution to be decoded $V := \{1, 2, ..., n\}$, where n is the number of vessels.
Let B be a set of m berths $B := \{1, 2, ..., m\}$.
Let L be a set of available berths for n vessels at low tide $L := \{L_1, L_2, ..., L_n\}$. //as explained in Section 3, L_j means available berths for vessel j are L_j, L_j+1, ..., m.
Let H be a set of available berths for n vessels at high tide $H := \{H_1, H_2, ..., H_n\}$. //$H_j$ means at high tide available berths for vessel j are H_j, H_j+1, ..., m.
Let the list $X_b := \{x_1, x_2, ..., x_k\}$ denote the sequence of vessels assigned to berth b, where k is length of sequence. Initially X_b is empty.
Let t_j denote the start time to serve vessel j.
Denote $proc_j$ the process time of vessel j, arr_j the arrival time of vessel j and w_j the weight of vessel j.
While Not all vessels have been scheduled **do**
T = 0; //at low tide
For $j := 1$ to n **do**:
For $b := L_j$ to m **do**:
Calculate the total cost at berth b $Cost_b = \sum_{n=1}^{k}(t_{x_n} - arr_{x_n} + proc_{x_n}) * w_{x_n}$.
For each position p ($1 \leq$ p $\leq$ k+1), calculate the increment of the objective value if vessel V_j is inserted to X_b. For example, if p = 1, the new sequence of vessels will be $\{V_j, x_1, ..., x_k\}$.
Update the start time of each vessel, such as $t_{V_j} = max\{0, arr_{V_j}\}$, $t_{x_1} = max\{t_{V_j} + proc_{V_j}, arr_{x_1}\}$ and etc.
Calculate the new cost $newCost_b = (t_{V_j} - arr_{V_j} + proc_{V_j}) * w_{V_j} + \sum_{n=1}^{k}(t_{x_n} - arr_{x_n} + proc_{x_n}) * w_{x_n}$.
Calculate the increment of the objective value i_p if current vessel is inserted to position p $i_p = newCost_b - Cost_b$.
Let $I_{b,V_j} := min_{p=1,2,...,k+1}\{i_p\}$ and $p_b := argmin_{p=1,2,...,k+1}\{i_p\}$.
End
Let $b' := argmin_{b=L_j, L_j+1, ..., m}\{I_{b,V_j}\}$. Insert V_j to position $p_{b'}$ of berth b'.
End
Remove the vessel from current schedule if its start time is later than the time of the current tide ends. For example, if vessel j has been sent to berth b, remove j from the list X_b.
T = T + TF; //at high tide
For $j := 1$ to n **do**:
For $b := H_j$ to m **do**:
Same as at low tide.
End
Same as at low tide.
End
Remove the vessel from current schedule if its start time is later than the time of the current tide ends.
Remove vessels which will finish in the next tide and the allocated berth is not available for this vessel in the next tide.
T = T + TF;
End

Algorithm 4 The pseudocode of *Local_search()*

Let S be the current schedule.
Let B be a set of m berths $B := \{1, 2, ..., m\}$.
//swap in berth
For i:= 1 to m **do**:
  Let $A := S_{B_i}$ denote the list of vessels sent to B_i.
  For q:= 1 to SizeOf(A)-1 **do**:
    For w:= $q+1$ to SizeOf(A) **do**:
      Swap A_q and A_w and denote the new list $A^{'}_{q,w}$.
      Denote $I_{q,w}$ as the new objective value of $A^{'}_{q,w}$.
    End
  End
  Find the smallest cost $I_{q',w'}$ from I.
  If $I_{q',w'}$ is smaller than the original cost, swap vessel $q^{'}$ and $w^{'}$. S_{B_i}= $A^{'}_{q',w'}$.
End

//swap between berths
Let N be the number of times doing swapping between berths
For i:= 1 to m-1 **do**:
  Let $V_{B_i,q}$ denote a randomly chosen vessel with the position q in S_{B_i}.
  For j:= $i+1$ to m **do**:
    Let $V_{B_j,w}$ denote a randomly chosen vessel with the position w in S_{B_j}.
    Swap $V_{B_i,q}$ and $V_{B_j,w}$. Then check if B_i is available for $V_{B_j,w}$ and if B_j is available for $V_{B_i,q}$.
    If the new cost after the swap is lower, update S.
  End
End

//move vessels from one berth to another
For i:= m to 1 **do**:
  For j:= $i-1$ to 1 **do**:
    Let $V_{B_i,q}$ denote a randomly chosen vessel with the position q in S_{B_i}. Let $A := S_{B_j}$ denote the list of vessels sent to B_j, and w denote a randomly picked position in A.
    Check if B_j is available for $V_{B_i,q}$ and calculate the new cost.
    If the new cost is lower than before, insert $V_{B_i,q}$ to A_w.
  End
End

improved berth schedule with a smaller cost, the newly created berth schedule will be kept.

5. Computational experiments

In this section, we assess the performance of the proposed LF-BAP by carrying out a number of different experiments. All the algorithms are coded in Java and all the experiments are conducted on a PC with an Intel i7 (3.60 GHz) processor and 16GB RAM under Windows 7. In addition, the exact method is implemented by using the state-of-the-art commercial solver CPLEX 12.7 with a maximum execution time of 1 hour for each instance. 52GB maximum java heap size is set for each run by CPLEX.

In this work, four sets of problem instances are tested. Set I corresponds to instances used in [4, 5] with a maximum of 8 berths and 24 vessels. In the work of [5], the authors extended the problem instances up to 8 berths and 50 vessels (Set II). However, in reality a terminal can be even busier than the situation in Set I and II. It is more common now to see big ports handling more than 100 or 200 vessels per day. According to [35], Europe's two biggest ports in Antwerp and Rotterdam already carried over 400 ships per day in 2008. It was also mentioned in [36] (2016) that Port of Rotterdam handles averagely 383 vessels per day. Asian ports can be even busier [37]. Port of Singapore is used to be one of the world's busiest ports, receiving an average of 140,000 vessels on an annual basis (approximately 380 vessels per day) as of 2013 [38]. However, according to recent ranking tables, many Chinese ports are deemed larger nowadays [37, 39]. Live vessel trackers [40] also indicate that many Chinese ports, such as Shanghai, Zhoushan, Qingdao, Ningbo, Tianjin, have up to dozens of hundreds vessels in ports, and a few hundred expected vessels at a moment in time. This indicates a likely turnaround of hundreds of vessels per day in those ports. Elsewhere in other continents, there are also other ports with more than 100 vessels per day, e.g. Houston [41], Tubaran [42], Cartagena [43], etc.

In addition, the problem considered in this paper deals with multiple tidal windows (there is no limit on the number of tidal windows to be considered). This makes it highly possible for ports to consider a berth schedule for multiple days, increasing the number of vessels per instance. It means ports with only a few dozen vessels per day (very

common in the real world) can still have large scale instances with more than 100 vessels. This shows the practicability of LF-BAP. Therefore, we believe conducting experiments on large-scale scenarios is meaningful for studying berth planning problems in real world. In order to simulate challenging real-world problems, we generate large-scale data sets III and IV following the instruction in [4]. Set III extends Set II to 50 berths and 500 vessels and Set IV represents extremely busy terminals in small - medium size with maximum 10 berths and 300 vessels. TF = 12 hours in every instance and they all start from low tide. An example instance is listed in Table 3 and a corresponding feasible solution is displayed (Fig. 1).

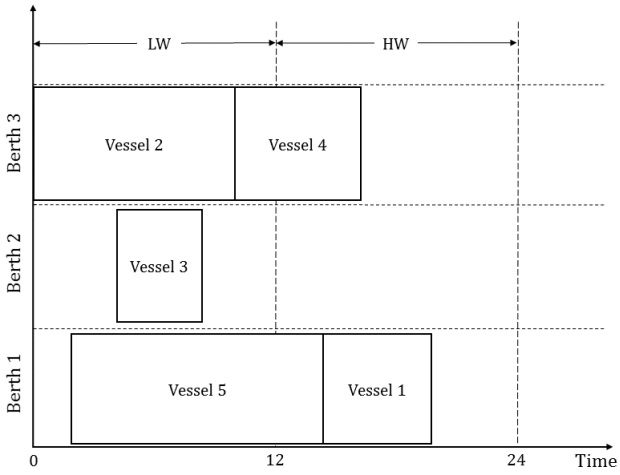


Figure 1: A feasible solution of the example instance given in Table 3

5.1. Comparing with an exact method and heuristics

5.1.1. A state-of-the-art exact method

In optimisation problems, exact algorithms are designed in a way that they guarantee finding an optimal solution in a finite amount of time. This finite amount of time usually grows with the problem size. Since BAPs are NP-hard [6, 7], exact methods may need exponential effort for even medium-sized problems. For example, for each vessel j, we may need a binary variable $S_{j,b}$ denoting if vessel j is sent to berth b. Assume there are a set of vessels V scheduled to berth b, we also need to denote another binary variable $P_{i,j}$ to indicate if vessel i will be berthed before vessel j, where i, j belong to V. This leads to a large number of combinations of integer values for the variables that must be tested. If the problem size increases, the complexity of the problem is highly affected and so the number of such combinations will rise dramatically.

In the experiments, we compare the performance of the proposed LF-BAP with a state-of-the-art exact method called Generalised Set-Partitioning BAP [5] with a multi-period planning time horizon (CPLEX-BAP). Based on the reported literature, CPLEX-BAP is one of the few publications considering tidal constraints. As an exact method it has shown the capability of solving small and medium scale problems with a reasonable running time. By conducting the experiments in this section, we will have some insights of how our algorithm performs comparing to the exact method in terms of the running time and objective values while the complexity of the problem increases.

Table 3: An example instance using notations from Section 3.2

Vessel j	t_arr_j	t_proc_j	w_j	L_j	H_j
1	12	5	1	1	1
2	0	10	2	3	1
3	5	3	8	2	1
4	10	6	5	3	3
5	2	12	4	3	2

5.1.2. A modified heuristic and a modified Iterated Greedy heuristic

In general, heuristics can provide solutions quickly like the greedy heuristic [4]. [4] prioritises vessels based on the available berths. Vessels with the same number of available berths are grouped together. Vessels in each group are sorted by the weighted processing time t_proc_j/w_j. And then vessels are sent to the schedule one by one following certain rules. A solution is obtained quickly because heuristic algorithms like [4] follow preset and heuristic rules that can be computed rapidly without any stochastic exploration. The downside of this heuristic could be that it is deterministic and hence is prone to always converging at a local optimum.

A comparison between LF-BAP and a modified heuristic (H-BAP) from [4] would be able to show whether stochastic elements in LF-BAP significantly improve the results. H-BAP repeats the algorithm in [4] so that it fits multiple tidal win-

dows. In the original algorithm in [4], the second tidal period goes to infinity which is not feasible for real-world problems. With the modification explained in 4.1.3, it guarantees feasible solutions to have a fair comparison with our work.

We also compare LF-BAP with an Iterated Greedy heuristic (IG) which is one of the best single-solution based meta-heuristics for BAPs in [44]. The initial solution is generated using the first-come-first-serve rule. In each iteration, IG improves the solution by reconstructing it with a greedy method. The reconstruction phase randomly chooses a number of vessels and reinserts them to their best positions. The evaluation function of IG is modified to fit the tidal constraints. With the comparison of LF-BAP and the single-solution based meta-heurstic IG, the effectiveness of the proposed method will be validated. In addition, in Section 5.2 we run experiments on a population-based meta-heuristic PSO [15] in order to show the features of population-based meta-heuristics on solving BAPs.

5.1.3. Sensitivity analysis of LF-BAP

With the goal of studying the impact of parameter settings in LF-BAP, a statistical analysis is conducted in this subsection. In Levy flight distribution, when the parameter β increases the frequency of having long jumps decreases. Because the distribution is heavy tailed, the increase of β would not make much difference if the value gets too big. l_0 controls the overall scale of jumps so it is preferred to be not too big because too many swaps in one iteration slows down the whole process. Ensuring l_0 greater than 1 significantly decreases the probability of having a jump distance less than 1. Therefore, parameter values are chosen from l_0 = 1, 3, 5, 8, 10 and β = 0.5, 1.5, 3, 5.

Nine random instances are chosen in different sizes among Set I, II, III and IV. A Friedman test shows that there is no significant difference of objective values between different parameter settings once the algorithm is converged. We then compare the runtime of different settings by allowing the algorithm to run until it converges to the same objective value. A significant difference appears after conducting the Friedman statistic test for all the combinations of l_0 and β. We notice a much larger runtime when β = 0.5 (Fig. 2). Excluding the combinations with β = 0.5, another Friedman test is conducted for all the other groups. The p-value $>$ 0.1 is achieved, so the null hypothesis of equality of treatments is accepted at 95% and 99% confidence. It means there is no significant differences between results from each group in terms of runtime. Thus, the performance of LF-BAP is not noticeably sensitive to all the groups except for β = 0.5. For the best overall performance, l_0 = 10 and β = 3 are selected for the following comparison with other algorithms.

Table 4: Parameter settings

Algorithms	Population size	Other parameters
LF-BAP	1	Maximum iterations = 10000; l_0 = 10; β = 3.
CPLEX-BAP	Not applicable	Time limitation: 1 hour. 52GB heap size. Other parameters set as default.
IG	1	Maximum iterations = 10000; α_{min} = 4; α_{max} = 7.
PSO	20	Maximum iterations = 500; C_1 = 2; C_2 = 2; W = 0.5.

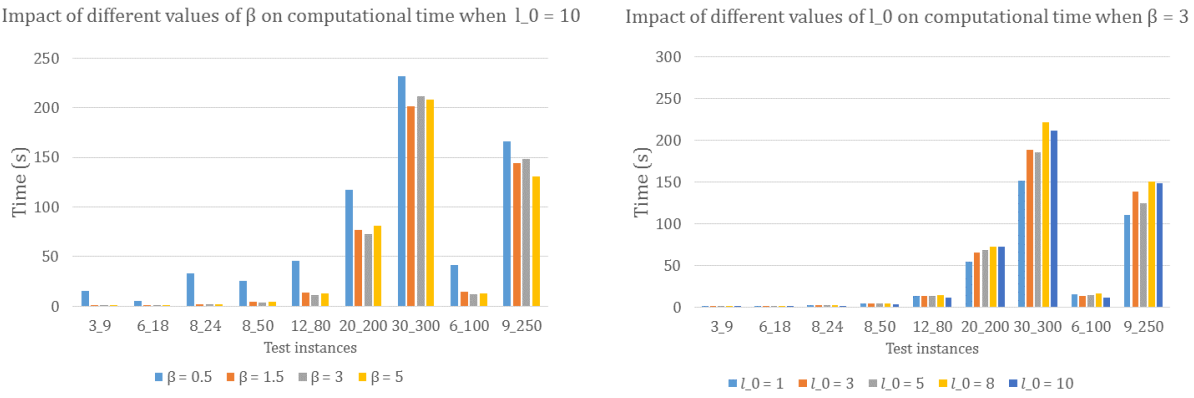
We also study the speed of convergence in terms of solution quality, i.e. objective value (Fig. 3). Since all the randomly selected instances show a similar pattern of convergence, two representative test instances are displayed. From the plots in Fig. 3, we have the following observations:

- Fig. 3(a) and 3(b) report the impact of l_0 with the fixed β = 3. A slower convergence is observed when l_0 = 1 for almost all the tested cases. Due to the small scale of jumps when l_0 = 1, the algorithm takes longer to search for a better solution. Other l_0 values yield similar convergence speed.
- None of the β values shows a significant difference in convergence speed when l_0 = 10. However, when β = 0.5 the algorithm converges slightly slower than the others in medium to large scale cases. This observation has also been pointed out in the above sensitivity analysis on computational time.

The parameter setting of LF-BAP in the following experiments after the sensitivity analysis is shown in Table 4. All the settings of other algorithms are shown in the table as well.

5.1.4. Computational results

The experiments evaluate the performance of our algorithm in terms of accuracy, efficiency and capability. We run LF-BAP and IG 50 times for each instance and one execution for CPLEX-BAP and



(a) (b)

Figure 2: Figures show the impact of different parameter settings on computaitonal time with fixed l_0 and β, respectively.

Table 5: Summary of the comparison between LF-BAP and CPLEX-BAP [5]

Data set	No. of test cases	Solvable cases		Faster algorithm		Percentage of cases with an error (%) e between LF-BAP and CPLEX-BAP				
		CPLEX-BAP	LF-BAP	CPLEX-BAP	LF-BAP	$e \leq 0.5\%$ or LF-BAP is better	$0.5\% < e \leq 1\%$	$1\% < e \leq 2\%$	$2\% < e \leq 5\%$	$e > 5\%$
I	120	120	120	47	73	66.67%	11.67%	10.83%	5.83%	5.00%
II	30	30	30	0	30	30.00%	16.67%	13.33%	30.00%	10.00%
III	100	40	100	0	100	60.00%	0.00%	10.00%	27.00%	3.00%
IV	60	20	60	0	60	66.67%	0.00%	0.00%	10.00%	23.33%
Total	310	210	310	47	263	60.97%	6.13%	8.7%	15.81%	8.39%

H-BAP because these two methods will always find the same solutions. Firstly, the accuracy of our work is discussed based on the objective values, the number of instances that LF-BAP is able to find the global optima, and the relative error. A relative error of LF-BAP (12) is defined as the gap between the mathematically proven global optima (found by the exact technique in CPLEX_BAP) and LF-BAP. The gap between LF-BAP and IG is also measured in following tables. The number can be negative which means LF-BAP outperforms the peer algorithms. Secondly, the running time represents how quickly and efficiently an algorithm reaches its optimal solution. Finally, we evaluate the capability of accommodating algorithms in large-scale problems.

$$e = \frac{Avg.Obj_{LF} - Avg.Obj_{CPLEX}}{Avg.Obj_{CPLEX}} * 100\% \quad (12)$$

Accuracy In Table 5, if CPLEX-BAP cannot solve the problem due to the out-of-memory error while LF-BAP can solve it, we consider LF-BAP is the faster algorithm and more accurate than CPLEX-BAP. According to Table 5, there are 91.61% of the test cases where LF-BAP found the global optima with an error $e \leq 5\%$, of which 60.97% cases has an error $e \leq 0.5\%$. For problems in Set I, most of the average relative error are less than 1%. LF-BAP provides similar quality results (about 1% error) for solving most of test cases in Set II, except for two instances with errors of 4.44% and 6.85%. In the results of large-scale problems (Table 7 and 8), the relative errors between LF-BAP and CPLEX-BAP become a bit larger for the instances CPLEX-BAP can solve. The overall percentage with an error less or equal than 5% is still above 60% for Set III and IV although it should be noted that in these sets there are only few instances where errors can be determined thanks to CPLEX-BAP being able to solve them to optimality. As an exact optimisation method, CPLEX-BAP guarantees an optimal solution if the problem size is small. Since both discrete and continuous BAPs have been proved to be NP-hard [6, 7], exact methods need exponential efforts to solve it when the problem scale increases [45]. It is reflected in Table 7 and 8 where the computational time of CPLEX-BAP increases exponentially in solving medium to large scale problems, and most of the cases become intractable. On the other hand, being an approximate method, LF-BAP can fare bet-

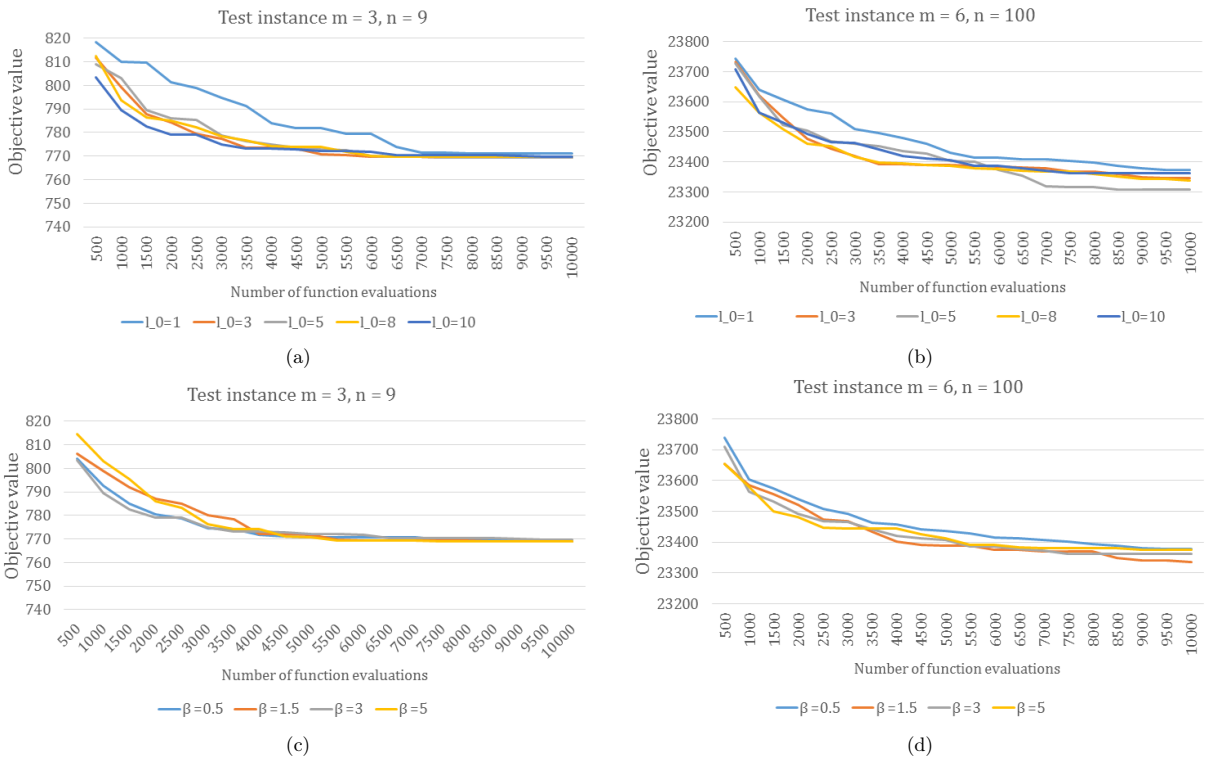


Figure 3: Figures show the impact of different parameter settings on objective values with fixed l_0 and β, respectively.

ter in large scale problems because it offers a better trade-off between solution quality and computational time. While exact methods put demands on complex formulations of an optimisation problem, meta-heuristics are flexible to be adapted to a specific problem. LF-BAP however does not guarantee optimal solutions, so it does not perform the same as the exact method CPLEX-BAP. However, 94% of small-scale instances, LF-BAP still provides similar quality solutions ($e \leq 5\%$).

In comparison with H-BAP and IG, LF-BAP significantly outperforms H-BAP in all test cases in terms of objective values and relative errors (Table 6, 7 and 8). All the negative percentages in these three tables indicate that LF-BAP obtains much better solutions than H-BAP. It also shows a better performance than IG in almost all cases except for the smallest four instances in Table 6. All the other negative percentages in the last column of Table 6, 7 and 8 indicate that LF-BAP obtains much better solutions than IG, especially when the problem scale gets large.

Table 6: A comparison of average objective values and average computational time between H-BAP, IG, CPLEX-BAP and LF-BAP on instances Set I and II. Avg states the average value of all the test cases in each problem size. In Set I, each problem size (each row) contains 10 cases and 5 cases in Set II. UB indicates the initial upper bound found by CPLEX-BAP.

Set	Problem size	Tidal effect	H-BAP [4]		IG [44]		CPLEX-BAP [5]				LF-BAP		Error between LF-BAP and CPLEX-BAP (%)	Error between LF-BAP and H-BAP (%)	Error between LF-BAP and IG (%)
			Avg. Obj.	Avg. Time (s)	Avg. Obj.	Avg. Time (s)	Avg. Obj.	Avg. Time (s)	Avg.UB Obj.	Avg.UB Time (s)	Avg. Obj.	Avg. Time (s)			
I	m=3, n=9	small	535.5	0.0043	467.3	0.0038	467	0.0872	1025	0.0022	469.8	0.0487	0.59%	-12.28%	0.75%
	m=3, n=9	big	666.7	0.003	573.7	0.0038	573.7	0.0427	1257	0.0001	587	0.0690	2.32%	-11.95%	1.85%
	m=4, n=12	small	791.3	0.0036	718.4	0.0627	717.4	0.101	2013.3	0.006	722.2	0.0510	0.67%	-8.73%	0.62%
	m=4, n=12	big	811.7	0.003	717.2	0.0787	712.5	0.0926	2158.1	0.006	718.8	0.0964	0.88%	-11.45%	0.22%
	m=5, n=15	small	958.5	0.003	868.3	0.0939	863.3	0.2814	2935.2	0.1324	869.5	0.2680	0.72%	-9.28%	-0.02%
	m=5, n=15	big	1066	0.0035	959.2	0.1303	947.3	0.2419	2824.4	0.1109	954.1	0.3906	0.71%	-10.50%	-0.46%
	m=6, n=18	small	1081	0.0036	982.6	0.1591	970.1	0.5698	3380.8	0.3738	973.3	0.4030	0.33%	-9.97%	-0.92%
	m=6, n=18	big	1215.2	0.003	1083.3	0.0647	1057.1	0.5939	3660.4	0.3749	1072.5	0.5715	1.46%	-11.74%	-0.81%
	m=7, n=21	small	1416.4	0.0031	1308	0.1726	1279.8	1.2425	4179.7	0.9485	1286.3	0.7554	0.51%	-9.19%	-1.56%
	m=7, n=21	big	1384.9	0.003	1238.8	0.1764	1195.6	1.2698	4355.9	0.0001	1201.8	1.2521	0.52%	-13.22%	-2.97%
	m=8, n=24	small	1577.2	0.003	1461.1	0.1796	1420	2.4369	5347.6	1.9069	1426.3	1.5097	0.44%	-9.57%	-2.39%
	m=8, n=24	big	1623.1	0.003	1455.2	0.1854	1394	2.1944	4561.8	1.7904	1411.4	1.8055	1.25%	-13.04%	-2.9%
II	m=6, n=30	small	1470.8	0.0038	1304.1	0.2898	1210.8	4.5912	7876.6	3.9872	1221.7	0.8000	0.90%	-16.94%	-5.98%
	m=6, n=30	big	1606.6	0.0106	1366.1	0.3208	1277.4	6.1932	7679	5.5132	1294	1.2092	1.30%	-19.45%	-5.13%
	m=7, n=40	small	3474	0.0046	3038.2	0.384	2808.4	24.6882	14017.4	22.8302	2841.4	2.7945	1.18%	-18.21%	-5.58%
	m=7, n=40	big	2965	0.0062	2408.9	0.425	2155.8	20.3352	12863.2	18.9092	2251.5	3.1968	4.44%	-24.06%	-6.35%
	m=8, n=50	small	2713.2	0.0064	2541.7	0.4801	2276.8	51.958	18654.8	47.954	2300.6	4.2861	1.05%	-15.21%	-9.19%
	m=8, n=50	big	4162.6	0.0046	3523.1	0.4438	3039	41.2578	15104	37.9658	3247.1	4.7728	6.85%	-21.99%	-8.16%

Table 7: Computational results on large scale data Set III. In Set III, each problem size (each row) contains 5 cases.

Set	Problem size	Tidal effect	H-BAP [4]		IG [44]		CPLEX-BAP [5]				LF-BAP		Error between LF-BAP and CPLEX-BAP (%)	Error between LF-BAP and H-BAP (%)	Error between LF-BAP and IG (%)
			Avg. Obj.	Avg. Time (s)	Avg. Obj.	Avg. Time (s)	Avg. Obj.	Avg. Time (s)	Avg.UB Obj.	Avg.UB Time (s)	Avg. Obj.	Avg. Time (s)			
III	m=9, n=60	small	6924.6	0.014	6390.5	0.4973	5746.2	57.1016	19994.6	49.4976	5847.4	5.0637	1.76%	-15.56%	-8.48%
	m=9, n=60	big	6676.2	0.0052	6459.8	0.5933	5603.6	52.7044	20582.2	44.1524	5798.8	6.0131	3.48%	-13.14%	-10.18%
	m=10, n=70	small	8383.6	0.0086	8260.6	0.6061	7257.2	112.3554	24984.2	93.2554	7473.8	7.0482	2.98%	-10.85%	-9.48%
	m=10, n=70	big	8233.2	0.0058	7905.9	0.5579	6704.6	107.0084	27255.8	78.7364	6957.4	7.9936	3.77%	-15.50%	-11.88%
	m=12, n=80	small	8965.4	0.004	8629.8	0.7014	7724.8	673.6474	27838	665.5674	7911.6	11.0984	2.42%	-11.75%	-8.06%
	m=12, n=80	big	8986	0.0042	8907.6	0.6501	7589.4	2761.022	30329.2	2736.898	7901.9	11.7207	4.12%	-12.06%	-11.31%
	m=13, n=100	small	12262.6	0.0062	11718.2	0.8322	10325.6	3577.88	38603.8	3493.904	10599	14.1122	2.65%	-13.57%	-9.55%
	m=13, n=100	big	11673	0.0054	11789.2	0.8185	9977	2713.847	40098.4	2640.987	10316.7	16.2133	3.40%	-11.62%	-12.47%
	m=14, n=120	small	15581.2	0.0092	15335.9	1.1342	N/S	N/S	N/S	N/S	13835.1	25.1970	N/S	-11.21%	-9.77%
	m=14, n=120	big	16724	0.0094	16554.5	1.1765	N/S	N/S	N/S	N/S	14421	28.0820	N/S	-13.77%	-12.94%
	m=15, n=150	small	23094.8	0.0076	22515	1.4783	N/S	N/S	N/S	N/S	19904.8	35.6046	N/S	-13.81%	-11.57%
	m=15, n=150	big	23457.8	0.0166	23722.6	1.4057	N/S	N/S	N/S	N/S	20215.8	38.9161	N/S	-13.82%	-14.67%
	m=20, n=200	small	30812.4	0.018	30682.3	1.7822	N/S	N/S	N/S	N/S	27116.1	69.4433	N/S	-12.00%	-11.57%
	m=20, n=200	big	30874.2	0.0102	30678.4	2.032	N/S	N/S	N/S	N/S	26182.2	75.6344	N/S	-15.20%	-14.62%
	m=30, n=300	small	45951.6	0.0166	44092.2	2.669	N/S	N/S	N/S	N/S	39529.1	214.9714	N/S	-13.98%	-10.35%
	m=30, n=300	big	47400.6	0.0152	46853.7	3.3009	N/S	N/S	N/S	N/S	40281.8	215.5681	N/S	-15.02%	-14.01%
	m=40, n=400	small	61800.8	0.022	59499	3.3434	N/S	N/S	N/S	N/S	52936.2	468.3557	N/S	-14.34%	-10.94%
	m=40, n=400	big	60219.6	0.0216	60061.3	4.4109	N/S	N/S	N/S	N/S	51751.4	438.4603	N/S	-14.06%	-13.85%
	m=50, n=500	small	77196.8	0.0316	73448.2	5.2914	N/S	N/S	N/S	N/S	66371.4	652.9757	N/S	-14.02%	-9.63%
	m=50, n=500	big	77988.8	0.0322	75601.8	5.4872	N/S	N/S	N/S	N/S	65460.2	955.2202	N/S	-16.06%	-13.4%

N/S: Not solvable by CPLEX-BAP.

Table 8: Computational results on large scale data Set IV. In Set IV, each problem size (each row) contains 5 cases.

Set	Problem size	Tidal effect	H-BAP [4]		IG [44]		CPLEX-BAP [5]				LF-BAP		Error between LF-BAP and CPLEX-BAP (%)	Error between LF-BAP and H-BAP (%)	Error between LF-BAP and IG (%)
			Avg. Obj.	Avg. Time (s)	Avg. Obj.	Avg. Time (s)	Avg. Obj.	Avg. Time (s)	Avg.UB Obj.	Avg.UB Time (s)	Avg. Obj.	Avg. Time (s)			
IV	m=5, n=80	small	20560.8	0.0208	20327.1	1.03	17533.6	71.4994	54128.2	30.6754	18295.3	6.6913	4.34%	-11.02%	-9.97%
	m=5, n=80	big	22354.2	0.007	21422.4	1.0704	18048.2	58.1214	55926.2	32.5494	19465.6	9.2297	7.85%	-12.92%	-9.17%
	m=6, n=100	small	25114.4	0.0072	25213	1.2498	21001.4	169.952	66660.2	103.3	22131.7	11.9623	5.38%	-11.88%	-12.2%
	m=6, n=100	big	27525.6	0.0064	27892.4	1.3613	22512	519.1268	78162.6	443.9588	24464.4	15.5042	8.67%	-11.12%	-12.27%
	m=7, n=150	small	47946	0.0116	48261.3	2.3068	N/S	N/S	N/S	N/S	41710.2	31.9941	N/S	-13.01%	-13.58%
	m=7, n=150	big	50884.6	0.012	52033.5	2.4503	N/S	N/S	N/S	N/S	44626.2	40.8227	N/S	-12.30%	-14.23%
	m=8, n=200	small	77732.4	0.0234	82519	3.2408	N/S	N/S	N/S	N/S	68726.4	67.1615	N/S	-11.59%	-16.68%
	m=8, n=200	big	78247.4	0.021	82894.3	3.2567	N/S	N/S	N/S	N/S	69696.7	77.2071	N/S	-10.93%	-15.86%
	m=9, n=250	small	113770.8	0.0294	118249.6	4.642	N/S	N/S	N/S	N/S	98179.3	123.0593	N/S	-13.70%	-16.89%
	m=9, n=250	big	108725.4	0.0322	113971	4.8733	N/S	N/S	N/S	N/S	95080.4	141.5667	N/S	-12.55%	-16.56%
	m=10, n=300	small	139798	0.0466	150286	5.1366	N/S	N/S	N/S	N/S	122812.5	199.9363	N/S	-12.15%	-18.16%
	m=10, n=300	big	141726.8	0.0488	152382.8	6.1128	N/S	N/S	N/S	N/S	124669	229.4605	N/S	-12.04%	-18.2%

N/S: Not solvable by CPLEX-BAP.

Efficiency All three algorithms start with a very small computational time (less than 1 second) according to Table 6. The meta-heuristic shows a much slower increase of computational time than the exact method when the problem size increases. (Fig. 4). To solve the largest problem size (m=8, n=50) in Set I and II, CPLEX-BAP takes 10 times of the running time of LF-BAP. This becomes more obvious in the large-scale test cases in Table 7, 8 and Fig. 4. The running time of CPLEX-BAP rises to about 3600 seconds on the largest instance it could solve while LF-BAP only needed about 14 seconds for that same instance. When the problem size grows up, the difference of running time between CPLEX-BAP and the meta-heuristic is significant (Fig. 4). We also report the initial upper bounds and the time CPLEX took to find them in Table 6, 7 and 8. There are some large-scale test cases where CPLEX- BAP could not find an upper bound. In these cases, CPLEX fails to complete the initialisation stage due to the out-of-memory error. For the rest, it takes CPLEX-BAP a significant amount of time to find the upper bounds. The time to find the initial upper bound is almost the same as the total time it takes to find the global optima. The quality of the initial upper bounds are significantly worse than the results by LF-BAP. This suggests that for this particular class of problem, CPLEX-BAP is slow to find an upper bound but it can then quickly converge to the optimal solutions.

Capability The largest problem CPLEX-BAP is able to solve in Set III and IV is 13*100 and 6*100, respectively. Table 5 displays that over 310 instances in total, CPLEX-BAP is able to find a solution for 210 instances while LF-BAP can find global optima for some instances and good sub-optimal solutions for all the rest. The coefficient of variation plot of LF-BAP shown in Fig. 5 represents the statistical robustness of our algorithm. As an approximate method, it is possible that the results of LF-BAP vary for the same test instance in 50 runs. The coefficient of variation (CV) is defined as the ratio of the standard deviation to the mean. A smaller CV value indicates the performance of the algorithm is more stable statistically. As shown in Fig. 5, the CV values are mostly less than 1%. A majority of them is close to 0% indicating a very small difference between 50 runs. With a normal distribution assumed on the data of CV, [0.163%, 0.203%] is achieved as 95% confidence interval [46] for the mean of CV. Therefore, a stable performance of our algorithm can be concluded due to the small interval of the mean of CV.

In summary, with the Levy flight distribution and the proposed local search, LF-BAP significantly improves the state-of-the-art heuristic H-BAP in terms of solution quality on all test cases. H-BAP achieves errors from 10%-37% while LF-BAP achieves errors from 0.44%-8.67%. Compared with the state-of-the-art single-point meta-heuristic IG, LF-BAP also achieves much better quality solutions in the majority of Set I and all the test cases in Set II, III and IV. In terms of computational cost, H-BAP and IG have a very quick turnaround time for all the instances (less than 10 second). It can be seen from Fig. 4 (b and d) that comparing to H-BAP, IG and CPLEX-BAP, the increase of runtime of LF-BAP is not significant when the problem complexity gets high. It increases from 0.05 to 4.77 seconds in Set I and II. For the cases CPLEX-BAP is able to solve, the runtime of LF-BAP increases to about 16.2 seconds while it takes CPLEX-BAP almost 3600 seconds. For the largest cases which CPLEX-BAP could not solve, it takes LF-BAP about 955 seconds. Comparing to the state-of-the-art exact method CPLEX-BAP, LF-BAP is also better in terms of computational time and feasible solutions in 85% of all 310 test cases (100% of large-scale cases). It is worth to mention that CPLEX-BAP [5] is the only method that can guarantee the global optimal solutions in the cases that it can solve. The exact model in CPLEX-BAP is remarkably effective and should be the default first choice for instances with about 80 vessels or less, unless the port operators want fast solutions within a few seconds or minutes in these similar scale cases. However, the gap between the proposed LF-BAP and CPLEX-BAP (less than 5% in 92% of the cases) is acceptable in practical scenarios. The faster computational time and the ability to find a good solution in the cases where CPLEX-BAP fails are the advantages of LF-BAP, making it a better option for large scale scenarios or any scenarios requiring a fast solution.

5.2. Comparison with population-based meta-heuristics

As explained in Section 4, population-based meta-heuristics have various shortcomings in dealing with BAPs. In this section we compare the proposed LF-BAP with an existing state-of-the-art

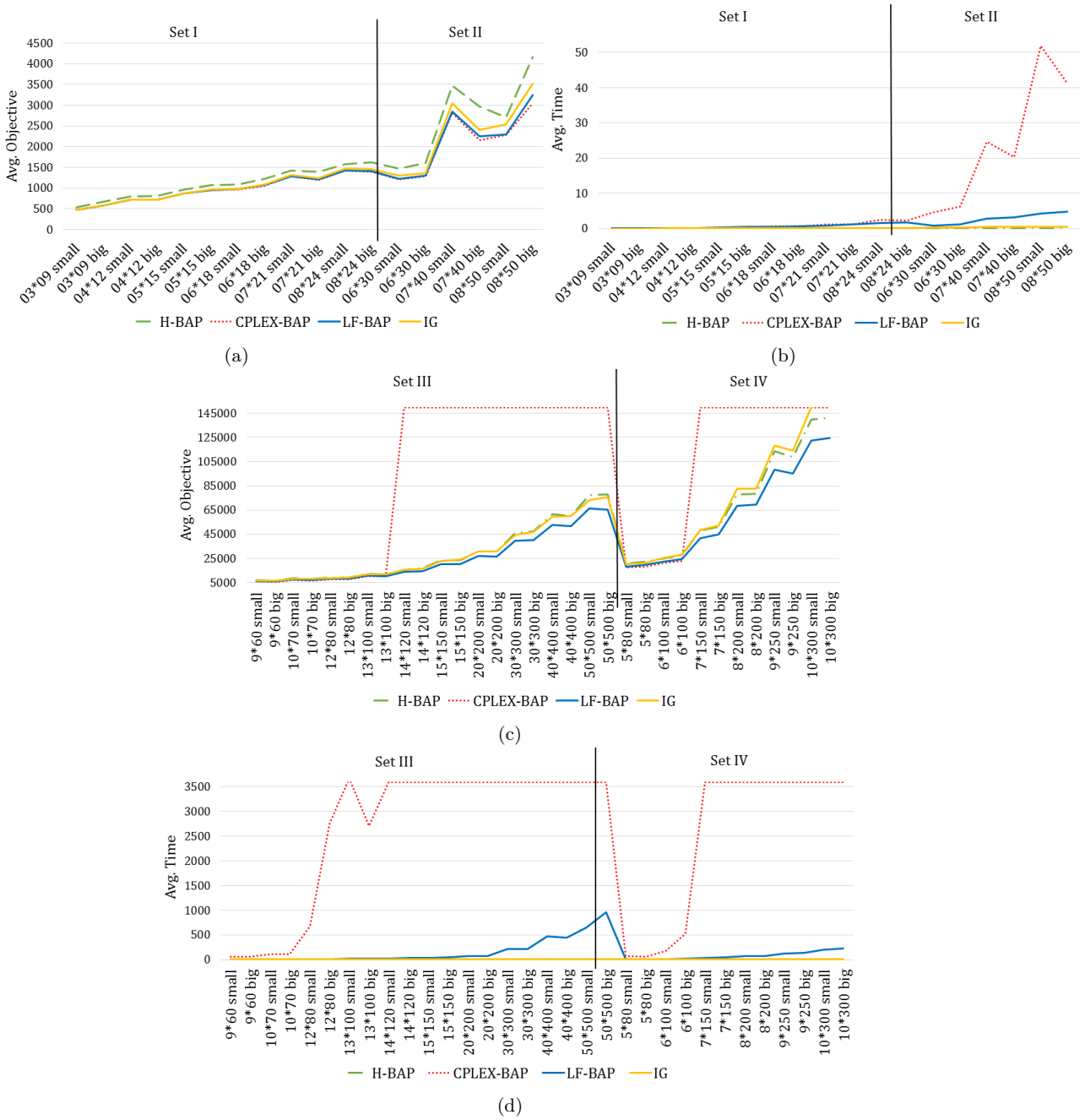


Figure 4: Comparison of four algorithms in terms of average running time and objective values. (a) Objective value comparison on small-scale data sets. (b) Runtime comparison on small-scale data sets. (c) Objective value comparison on large-scale data sets. CPLEX-BAP is given the maximum value on the cases it could not solve. (d) Runtime comparison on large-scale data sets. CPLEX-BAP is given the maximum value on the cases it could not solve.

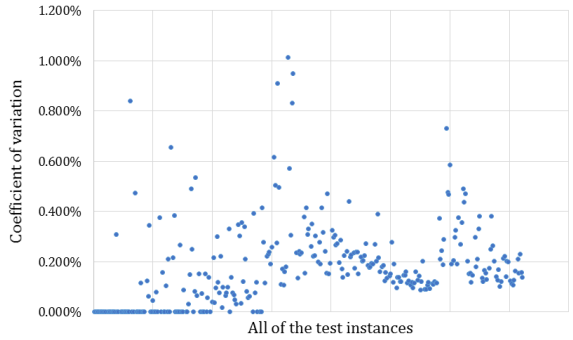


Figure 5: The coefficient of variation represents the robustness of LF-BAP.

PSO [15] for BAPs. We replicate the PSO from [15]. All details of the PSO remain the same as in [15] but we need to make some modifications to the evaluation function only to accommodate tidal constraints. The fitness function is the same as (1). When evaluating we check whether the berth allocated is available for every vessel. If there is no berth available for a particular vessel at any tide according to the given information, the solution (called particle in PSO) will be re-initialised. If a berth is only available at high tide, the vessel will wait to be scheduled until a high tide occurs.

We present here those instances with known global optima (found by CPLEX-BAP) so that the results can be compared more accurately. We can see from Table 9 that with the same number of function evaluations, LF-BAP significantly outperforms PSO in all of the test cases.

6. Conclusions

In this paper, we have proposed a new meta-heuristic based on Levy flight (LF-BAP) to solve the Berth Allocation Problem taking tidal effect into consideration. We have conducted several experiments comparing our algorithm with the state-of-the-art exact method from [5] and the heuristic from [4], a single-solution based meta-heuristic [44] as well as a population-based meta-heuristic from [15]. According to the results, LF-BAP is able to find highly competitive approximate solutions that are faster and feasible in large-scale cases. We believe the work we have demonstrated can make a practical contribution to the operations of seaports and terminals.

Table 9: Comparison between PSO and LF-BAP on some test instances

Problem size	Tidal effect	PSO [15]		LF-BAP	
		Avg. Obj.	Error (%)	Avg. Obj.	Error (%)
m=3, n=9	small	492.9	5.55%	469.8	0.59%
m=3, n=9	big	615.6	7.28%	587	2.32%
m=4, n=12	small	778.5	8.00%	722.2	0.67%
m=4, n=12	big	828.5	16.44%	718.8	0.88%
m=5, n=15	small	973.1	12.25%	869.5	0.72%
m=5, n=15	big	1113.0	18.10%	954.1	0.71%
m=6, n=18	small	1135.2	17.12%	973.3	0.33%
m=6, n=18	big	1311.3	24.27%	1072.5	1.46%
m=7, n=21	small	1551.9	22.47%	1286.3	0.51%
m=7, n=21	big	1544.5	28.43%	1201.8	0.52%
m=8, n=24	small	1819.7	27.87%	1426.3	0.44%
m=8, n=24	big	1872.7	33.94%	1411.4	1.25%
m=6, n=30	small	2452.2	103.20%	1221.7	0.90%
m=6, n=30	big	2544.7	110.06%	1294	1.30%
m=7, n=40	small	5772.6	111.99%	2841.4	1.18%
m=7, n=40	big	4938.0	127.74%	2251.5	4.44%
m=8, n=50	small	6744.3	189.77%	2300.6	1.05%
m=8, n=50	big	7960.5	181.51%	3247.1	6.85%
m=9, n=60	small	9816.1	70.37%	5847.4	1.76%
m=9, n=60	big	9779.7	76.21%	5798.8	3.48%
m=10, n=70	small	13502.3	84.11%	7473.8	2.98%
m=10, n=70	big	12555.5	86.65%	6957.4	3.77%
m=12, n=80	small	13620.9	71.90%	7911.6	2.42%
m=12, n=80	big	14025.8	85.01%	7901.9	4.12%
m=13, n=100	small	20189.5	94.30%	10599	2.65%
m=13, n=100	big	19568.9	97.79%	10316.7	3.40%
m=5, n=80	small	30819.5	77.97%	18295.3	4.34%
m=5, n=80	big	31039.9	71.36%	19465.6	7.85%
m=6, n=100	small	40559.0	92.75%	22131.7	5.38%
m=6, n=100	big	41507.1	87.17%	24464.4	8.67%

Error (%) comparing to CPLEX-BAP (12)

Acknowledgement

This work was supported by a Dean scholarship from the Faculty of Engineering and Technology, LJMU, a Newton Institutional Links grant no. 172734213, and a Newton Research Collaboration Programme grant, both from the UK BEIS. We also would like to thank Charly Lersteau for contributing the idea of statistical analysis.

References

[1] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, European Journal of Operational Research 202 (3) (2010) 615–627.

[2] C. Bierwirth, F. Meisel, A follow-up survey of berth allocation and quay crane scheduling problems in container terminals, European Journal of Operational Research 244 (3) (2015) 675–689.

[3] A. Imai, E. Nishimura, S. Papadimitriou, The dynamic berth allocation problem for a container port, Transportation Research Part B: Methodological 35 (4) (2001) 401–417.

[4] D. Xu, C.-L. Li, J. Y.-T. Leung, Berth allocation with time-dependent physical limitations on vessels, European Journal of Operational Research.
[5] E. Lalla-Ruiz, C. Expósito-Izquierdo, B. Melián-Batista, J. M. Moreno-Vega, A set-partitioning-based model for the berth allocation problem under time-dependent limitations, European Journal of Operational Research 250 (3) (2016) 1001–1012.
[6] A. Lim, The berth planning problem, Operations research letters 22 (2) (1998) 105–110.
[7] P. Hansen, C. Oguz, A note on formulations of the static and dynamic berth allocation problems, Citeseer, 2003.
[8] A. Imai, K. Nagaiwa, C. W. Tat, Efficient planning of berth allocation for container terminals in asia, Journal of advanced transportation 31 (1) (1997) 75–94.
[9] A. Imai, E. Nishimura, S. Papadimitriou, Berth allocation with service priority, Transportation Research Part B: Methodological 37 (5) (2003) 437–457.
[10] J.-F. Cordeau, G. Laporte, P. Legato, L. Moccia, Models and tabu search heuristics for the berth-allocation problem, Transportation science 39 (4) (2005) 526–538.
[11] E. Lalla-Ruiz, B. Melián-Batista, J. M. Moreno-Vega, Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem, Engineering Applications of Artificial Intelligence 25 (6) (2012) 1132–1141.
[12] P. Hansen, C. Oğuz, N. Mladenović, Variable neighborhood search for minimum cost berth allocation, European Journal of Operational Research 191 (3) (2008) 636–649.
[13] M. M. Golias, M. Boile, S. Theofanis, Berth scheduling by customer service differentiation: A multi-objective approach, Transportation Research Part E: Logistics and Transportation Review 45 (6) (2009) 878–892.
[14] R. M. de Oliveira, G. R. Mauri, L. A. N. Lorena, Clustering search for the berth allocation problem, Expert Systems with Applications 39 (5) (2012) 5499–5505.
[15] C.-J. Ting, K.-C. Wu, H. Chou, Particle swarm optimization algorithm for the berth allocation problem, Expert Systems with Applications 41 (4) (2014) 1543–1550.
[16] E. Lalla-Ruiz, J. L. González-Velarde, B. Melián-Batista, J. M. Moreno-Vega, Biased random key genetic algorithm for the tactical berth allocation problem, Applied Soft Computing 22 (2014) 60–76.
[17] L. Zhen, Tactical berth allocation under uncertainty, European Journal of Operational Research 247 (3) (2015) 928–944.
[18] L. Zhen, L. H. Lee, E. P. Chew, A decision model for berth allocation under uncertainty, European Journal of Operational Research 212 (1) (2011) 54–68.
[19] Y. Xu, Q. Chen, X. Quan, Robust berth scheduling with uncertain vessel delay and handling time, Annals of Operations Research 192 (1) (2012) 123–140.
[20] E. Lalla-Ruiz, S. Voß, C. Expósito-Izquierdo, B. Melián-Batista, J. M. Moreno-Vega, A popmusic-based approach for the berth allocation problem under time-dependent limitations, Annals of Operations Research (2015) 1–27.
[21] E. Lalla-Ruiz, S. Voss, Popmusic as a matheuristic for the berth allocation problem, Annals of Mathematics and Artificial Intelligence 76 (1-2) (2016) 173–189.
[22] É. D. Taillard, S. Voss, Popmusic partial optimization metaheuristic under special intensification conditions, in: Essays and surveys in metaheuristics, Springer, 2002, pp. 613–629.
[23] V. H. Barros, T. S. Costa, A. C. Oliveira, L. A. Lorena, Model and heuristic for berth allocation in tidal bulk ports with stock level constraints, Computers & Industrial Engineering 60 (4) (2011) 606–613.
[24] T. Nishi, T. Okura, E. Lalla-Ruiz, S. Voß, A dynamic programming-based matheuristic for the dynamic berth allocation problem, Annals of Operations Research (2017) 1–20.
[25] T. Qin, Y. Du, M. Sha, Evaluating the solution performance of ip and cp for berth allocation with time-varying water depth, Transportation Research Part E: Logistics and Transportation Review 87 (2016) 167–185.
[26] A. D. De León, E. Lalla-Ruiz, B. Melián-Batista, J. M. Moreno-Vega, A machine learning-based system for berth scheduling at bulk terminals, Expert Systems with Applications 87 (2017) 170–182.
[27] T. Tran, T. T. Nguyen, H. L. Nguyen, Global optimization using lévy flights, in: Proceedings of ICT.rda'04, Hanoi, 2004.
[28] Z. Meng, H. Shen, T. Zhao, Hybrid artificial bee colony algorithm based on cuckoo search strategy, in: Semantics, Knowledge and Grids (SKG), 2016 12th International Conference on, IEEE, 2016, pp. 136–140.
[29] X.-S. Yang, S. Deb, Cuckoo search via lévy flights, in: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, IEEE, 2009, pp. 210–214.
[30] G. Terdik, T. Gyires, Lévy flights and fractal modeling of internet traffic, IEEE/ACM Transactions on Networking 17 (1) (2009) 120–129.
[31] D. K. Sutantyo, S. Kernbach, P. Levi, V. A. Nepomnyashchikh, Multi-robot searching algorithm using lévy flight and artificial potential field, in: Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop on, IEEE, 2010, pp. 1–6.
[32] A. F. Ali, A hybrid gravitational search with levy flight for global numerical optimization, Information Sciences Letters Inf. Sci. Lett 4 (2015) 71–83.
[33] G. Viswanathan, E. Raposo, M. Da Luz, Lévy flights and superdiffusion in the context of biological encounters and random searches, Physics of Life Reviews 5 (3) (2008) 133–150.
[34] M. Gutowski, Levy flights as an underlying mechanism for global optimization algorithms, ArXiv Mathematical Physics e-printsarXiv:math-ph/0106003.
[35] NWEUROPE, Opportunities for Territorial Change (2008).
URL http://www.espace-project.org/publications/IIIBPublicationonBestIIIBprojects.pdf
[36] ForConstructionPros, Turning Sea Into Land (2016).
URL https://www.forconstructionpros.com/concrete/equipment-products/article/12278666/turning-sea-into-land
[37] Wikipedia, List of busiest container ports (2017).
URL https://en.wikipedia.org/wiki/List_of_busiest_container_ports
[38] Akanksha Gupta, The world's 10 biggest ports (2013).
URL http://www.ship-technology.com/features/feature-the-worlds-10-biggest-ports/
[39] World Shipping Council, TOP 50 WORLD CONTAINER PORTS (2015).
URL {http://www.worldshipping.org/about-the-industry/global-trade/

top-50-world-container-ports}
[40] Vesseltracker, Vesseltracker.com (2018).
URL http://www.vesseltracker.com/en/Home.html
[41] MarineLink, Collision closes houston ship channel (2015).
URL https://www.marinelink.com/news/collision-houston-channel387314
[42] Ta Kung Pao, Capesize Freight Route High (2011).
URL http://www.hh-ship.com/hh/en/newsshow.asp?id=35
[43] Andrew Mwaniki, Busiest Cargo Ports In South America (2018).
URL https://www.worldatlas.com/articles/busiest-cargo-ports-in-south-america.html
[44] S.-W. Lin, K.-C. Ying, S.-Y. Wan, Minimizing the total service time of discrete dynamic berth allocation problem by an iterated greedy heuristic, The Scientific World Journal 2014.
[45] G. J. Woeginger, Exact algorithms for np-hard problems: A survey, in: Combinatorial Optimization—Eureka, You Shrink!, Springer, 2003, pp. 185–207.
[46] J. Neyman, Outline of a theory of statistical estimation based on the classical theory of probability, Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences 236 (767) (1937) 333–380.
URL http://www.jstor.org/stable/91337