# BERTH SCHEDULING AT SEAPORTS: META-HEURISTICS AND SIMULATION

RAN WANG

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

October 2018

# Abstract

This research aims to develop realistic solutions to enhance the efficiency of port operations. By conducting a comprehensive literature review on logistic problems at seaports, some important gaps have been identified for the first time. The following contributions are made in order to close some of the existing gaps.

Firstly, this thesis identifies important realistic features which have not been well-studied in current academic research of berth planning. This thesis then aims to solve a discrete dynamic Berth allocation problem (BAP) while taking tidal constraints into account. As an important feature when dealing with realistic scheduling, changing tides have not been well-considered in BAPs. 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. We propose one single-point meta-heuristic and one population-based meta-heuristic. With our algorithms, 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. Comprehensive experiments are conducted in order to analyse the strengths and weaknesses of the algorithms and compare with both exact and approximate methods.

Furthermore, lacking tools for examining existing algorithms for different optimisation problems and simulating real-world scenarios is identified as another gap in this study. This thesis develops a discrete-event simulation framework. The framework is able to generate test cases for different problems and provide visualisations. With this framework, contributions include assessing the performance of different algorithms for optimisation problems and benchmarking optimisation problems.

# Acknowledgements

I would like to express my deepest and sincerest gratitude to my supervisor Dr. Trung Thanh Nguyen. Without his guidance, this PhD thesis would not have been possible. In the past three years, he has provided me with invaluable advice, constant encouragement and generous financial support. I feel extremely fortunate to work with him. He has always been open for discussion and has helped me with his knowledge and expertise as much as he could. I was given various great opportunities to deepen my research and experience in this field. I learned a lot from him, and I cannot thank him enough.

I also owe special thanks to my second supervisor Dr. Shayan Kavakeb for giving me so much helpful advice and generously sharing his experience. I gratefully appreciate his effort. My sincere thanks also go to Prof. Zaili Yang, Charly, Yannis, Daniel, Igor and Ahmed for their support and insightful suggestions.

I gratefully acknowledge the funding sources that supported my PhD study. I was funded by the Dean's Scholarship from the Faculty of Engineering and Technology, Liverpool John Moores University and the European EC-China Research Network on Integrated Container Supply Chains (ENRICH) project.

I also would like to thank my friends in Liverpool and in China for their support and company. Special thanks go to Dr. Rui Zhu for all her encouragement. I feel very lucky to have shared pleasant time with such a wonderful friend.

Lastly, no word can express my gratitude to my family, especially to my mum, who has given me everything. I would like to thank her for always supporting me and encouraging me to pursue what I want in my life. Without their unconditional love, I could not have completed this long journey.

This thesis is dedicated to my family.

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| **AGWO** | Adaptive Grey Wolf Optimiser |
| **ALNS** | Adaptive Large Neighbourhood Structure |
| **BAP** | Berth Allocation Problem |
| **BPP** | Bin Packing Problem |
| **CV** | Coefficient of Variation |
| **DBAP** | Dynamic Berth Allocation Problem |
| **FCFS** | First-Come-First-Serve |
| **GA** | Genetic Algorithm |
| **GNS** | Guided Neighbourhood Search |
| **GRASP** | Greedy Randomized Adaptive Search Procedure |
| **MA** | Memetic Algorithm |
| **POPMUSIC** | Partial Optimisation Meta-heuristics Under Special Intensification Conditions |
| **PSO** | Particle Swarm Optimisation |
| **QC** | Quay Crane |
| **SA** | Simulated Annealing |
| **SBAP** | Static Berth Allocation Problem |
| **SWO** | Squeaky Wheel Optimisation |
| **TS** | Tabu Search |
| **VND** | Variable Neighbourhood Descent |
| **VNS** | Variable Neighbourhood Search |
| **WSPT** | Weighted Shortest Processing Time |
| **2D** | Two-Dimensional |
| **3D** | Three-Dimensional |

# Chapter 1

# Introduction

The economic globalisation has been greatly boosting the amount of international trade. Container trade is one of the areas experiencing fast growth. The scale of import and export keeps expanding, hence, optimising the efficiency of port operations is vital for the mobility of goods. Applying optimisation techniques to container seaports has become an active research topic during the last few decades. Due to the wide variety of problems in port operations, current academic research may not have covered all the specific requirements of the industry. The main purpose of this thesis is to investigate some operational problems at ports, especially the berth planning problem, and to propose solutions to close some of the gaps.

## 1.1 Port operations and optimisation problems

Port operations are mainly categorised into areas such as berth, quay, transport area, storage yard, and terminal gate. Berth and quay are considered seaside, while storage yard and terminal gate are considered landside [Vis and De Koster, 2003]. In general, a transportation process at terminals starts from assigning vessels to berths. To do the unloading, a vessel has to moor at a berth and then quay cranes are used to unload containers from the vessel onto vehicles. Transport vehicles move containers from the quay to the storage yard which is an area to store containers temporarily. Containers at the storage yard will be transported to another vessel or imported to land according to the schedule. If containers are transported to another vessel, they will be moved by vehicles back to the seaside. If containers are going to be imported to land, they will be transported through the terminal gate.

According to UNCTAD 2015, the volume of container trade has shown a significant annual increase in recent decades. Due to the growing global maritime trade, the development of the supply chain network and the logistical operations has become even more challenging nowadays. The demand of loading and unloading cargos is significantly high. In reality, most of the time ports are not operating at 100% capacity and quays are under-utilised. For example, manual processes are still highly involved in berth scheduling. In order to continually adapt to the global supply chain, efficient port operations with up-to-date technologies are essential.

To link seaside and landside efficiently, a number of logistics optimisation problems in port operations have been studied relating to vehicles, vessels and loading and unloading. Recent trends and development in maritime transport and port operations with regards to optimisation problems have been summarised in Bierwirth and Meisel [2015]. Logistics optimisation problems have been widely studied in many fields such as transportation, scheduling, resource allocation etc. The level of complexity of this kind of problem is normally high when considering many constraints to simulate real-world scenarios. Optimisation methods frequently used to solve hard and complex problems include linear programming, branch and bound, evolutionary algorithms etc.

## 1.2 A case study of optimisation problems in port operations

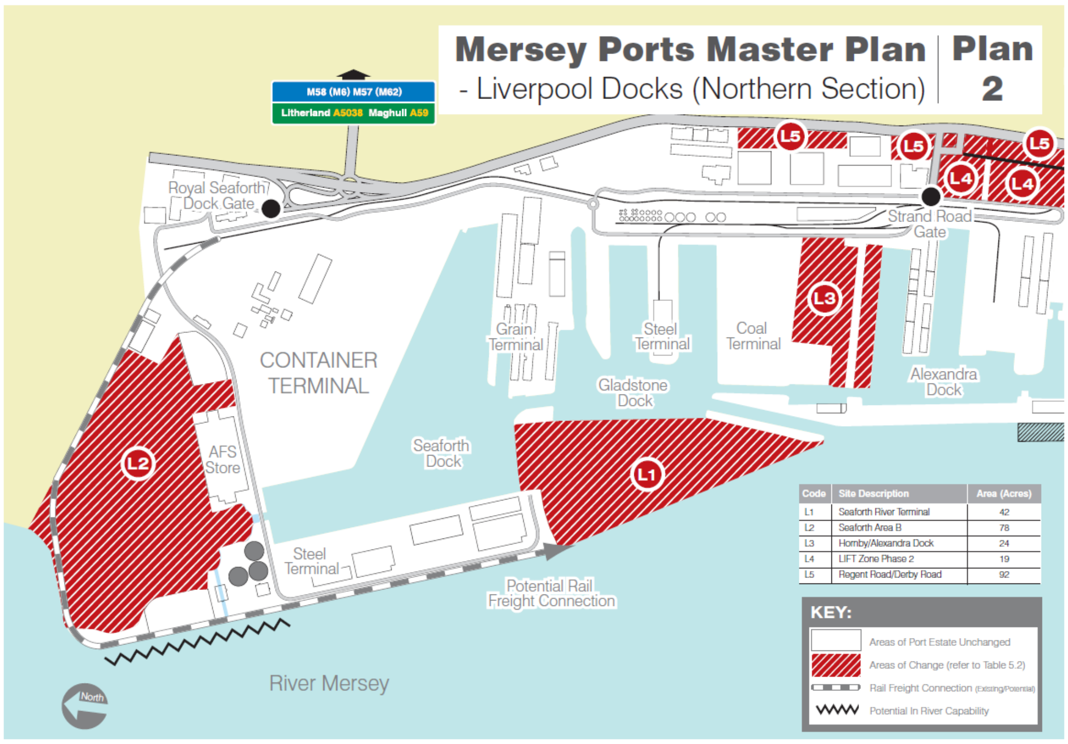
A real-world optimisation problem is explained in this section for demonstration: a berth allocation problem (BAP). The aim of BAPs is to schedule a set of vessels that are arriving at a terminal of a port. Before arrival, there are a number of factors to consider for berthing a vessel. The vessel information needed normally includes the measurements (the length and the draft), the expected arrival time, the handling time and necessary resources for loading and unloading.

In the Port of Liverpool, there are multiple docks for handling different vessels. A *dock* is an area of water that can be closed off and that is made for transferring cargoes. For the purpose of sending vessels to an area of water, the *lock* system is built. In a port, there are usually a number of locks for different destinations and specific purposes. If a vessel is entering a dock and the water level outside is different from that of the dock, the lock system is able to raise or lower the vessel in order to match the water level of the destination. Sometimes a vessel travelling outward through the lock can cause a decrease of the water level inside the dock. Some locking operations can only be done within certain water levels.

The tide is an influential factor of changing the water level. The height of the tide is cyclical. In the Port of Liverpool, the water level varies frequently due to the nature of fast changing tides. There are four tidal windows including two high tides and two low tides per day. When berthing at a dock and getting in and out of a lock, the water level at current tide has to be higher than the draft of the vessel. The complexity of this kind of berth planning is usually high if considering the changing of water levels. For such a busy port, failing to schedule the operations efficiently can generate a huge cost.

Furthermore, inside each lock there are multiple berths, each is suitable only to certain types of vessels due to their length, water depth, and availability of resources. For example, the Royal Seaforth terminal includes different berths suitable for different types of vessels carrying containers, oil, timber, fruit and vegetables, grain, animal food, and ferries go through the Gladstone lock (Fig. 1.1). The shipment of scrap metal and biomass power plants is sent to the Alexandra dock through the Langton lock. In order to manage the shipping in the Port of Liverpool, incoming vessels have to confirm the expected time of arrival with the terminal operator not later than ten days prior. The tides and details of shipment have to be agreed in advance as well.

In summary, multiple conditions need to be satisfied in order to arrange berths and resources for incoming vessels in real-world scenarios such as: 1) different types of vessels have to go to different berths; 2) the dimensions of vessels are also restricted since the draft available is subject to the height of tide, length of berths, available resources among other things; 3) the decision has to be made in advance according to the current schedule; 4) the schedule can be changed due to the uncertainty such as a delay of arrival. In the event of failure to optimise the schedule, the port and the shipowner may face a heavy penalty due to the time lost or work delayed.



| Code | Site Description | Area (Acres) |
|---|---|---|
| L1 | Seaforth River Terminal | 42 |
| L2 | Seaforth Area B | 78 |
| L3 | Hornby/Alexandra Dock | 24 |
| L4 | LIFT Zone Phase 2 | 19 |
| L5 | Regent Road/Derby Road | 92 |

Figure 1.1: An example of different terminals and locks in Port of Liverpool [Peel Ports Group, 2016].

## 1.3 Scope of the thesis

Because the range of optimisation problems at ports are wide and diverse, it is impossible to cover all the topics in this thesis. We will focus on BAPs and port simulations. In port operation planning, the BAP is considered a significant part of the planning. On the port side, an efficient schedule determines how many vessels a terminal is able to serve in a day and how much cost or benefit it will produce. On the fleet side, the schedule determines the time the vessel should arrive at the berth, which berth to arrive at and the associated cost. There are many factors influencing the whole scheduling procedure, such as the arrangement of resources, the availability of berths and the required departure time of vessels. The berth allocation can be the most expensive one out of all port operations, because if a vessel has to stay for longer (e.g. due to low tides or congestions), the shipping line may face a delay and an entire goods supply chain can be affected internationally. If a vessel cannot be admitted in certain time windows, the resources may be occupied over time and the port may have to pay heavy penalty depending on the contractual agreements.

Since real-world problems in port operations are always complex with stochastic elements, a simulation is often the solution to evaluate the feasibility and performance of an algorithmic solution. One of the major advantages of simulation at ports, is to investigate potential influences such as incremental cost and port congestion, of a development before actually applying it to a port. The proposed improvement can be monitored and estimated from a global perspective with simulations. Thus, it is also meaningful to investigate the port simulations in the thesis.

## 1.4 General research questions

The study in this thesis begins with some general questions. Through investigating these general research questions in the field of BAPs and port simulations, some definitions get clear and gaps are identified. With a clearer view of the field and the gap, specific research questions will be raised. The following parts will look for the answers of more specific questions. My general questions include:

*What are the gaps between academic BAPs and real-world scenarios? What is the current situation of port simulations in academic research? Is there any practical feature important but has not been studied well in academic research?*

In order to answer these questions, a comprehensive literature review is carried out. In Chapter 2, how BAPs and simulations have been addressed is investigated. Aspects in the literature that we look at include problem definitions, optimisation approaches, performance measures, benchmark problems and simulation applications. Therefore, we will have wide knowledge of this field and insights of the gaps. Once we have identified the gaps, more specific questions like below will be asked.

*If we have found some weaknesses in the problems, what are the most important ones that we should study and why?*

The solution to answer the question above can be converted to even more specific questions in terms of how to fill the gap. Further questions are raised as follows.

*How do we improve current situations? More importantly, how can we effectively solve these problems?*

In the rest of the thesis, our research will focus on finding the questions above and solving the problem to fill the gap.

## 1.5 Outline of the thesis

This thesis is an exploration following the research questions above. It is organised as follows: Chapter 2 reviews the related literature on BAPs and simulations at seaports. It studies: 1) what constraints and features are included in BAPs; 2) how these features have been addressed in optimisation approaches; 3) what optimisation techniques are used to solve them; 4) how the performance has been evaluated and what are the current benchmarks; 5) how simulations have been applied to ports and container terminals; 6)what optimisation problems have been integrated to port simulations. The purpose of this literature review is to identify the difference and gaps between academic models and real-world problems.

One of the gaps is identified in Chapter 2: the lack of consideration given to changing tides in berth planning problems in academic research. Tidal constraints are important for real-world problems but they have been barely investigated. In Chapters 3 and 4, we try to close the gaps by using optimisation techniques. We focus on improving the performance of the identified problem while considering important features in a real-world scenario. The reason for the lack of methods on this problem may be that the tidal constraints greatly increase the complexity of the model. One single-point meta-heuristic and one population-based meta-heuristic are proposed to solve the BAP with multiple tidal windows. We conduct several experiments in order to study both algorithms and the performance compared to other approaches.

In Chapter 5, we deal with other weaknesses that also are pointed out in Chapter 2 including: 1) the lack of a general framework/platform, 2) the difficulty of comparing different optimisation algorithms, and 3) the lack of visualised results. A framework is developed as a handy tool to tackle these difficulties. The structure of the framework is explained in detail. Moreover, BAPs and bin packing problems are shown as examples of integrating optimisation algorithms for different problems by using this framework. The implementation of how the framework accommodates a variety of bin packing problems with different uncertainties and how the test instances are generated is explained by providing flow charts and pseudo code. It also identifies performance measures in order to conduct efficient and fair comparison of multiple algorithms.

Chapter 6 concludes the work in the thesis and summarises the contributions. Future potential research directions are also suggested.

## 1.6 Articles resulting from this thesis

**Refereed or submitted journal papers**

1. Wang, R., Nguyen, T.T., Li, C, Jenkinson I, Kavakeb, S. and Yang, Z., 2017. Optimising discrete dynamic berth allocation in seaports using a Levy flight based metaheuristic. Revision submitted to Swarm and Evolutionary Computation.

**In-preparation journal paper**

2. Wang, R., et al., Z., A Genetic algorithm to solve the berth allocation problem with tidal windows. To be submitted in July, 2018.

**Refereed conference papers**

3. Wang, R., Nguyen, T.T., Kavakeb, S., Yang, Z. and Li, C., 2016, March. Benchmarking dynamic three-dimensional bin packing problems using discrete-event simulation. In European Conference on the Applications of Evolutionary Computation (pp. 266-279). Springer, Cham.

4. Wang, R., Nguyen, T.T., Kavakeb, S., Yang, Z. and Li, C., 2016, September. A simulation framework for benchmarking 3D bin packing problems under uncertainties. In LRN Proceedings 2016.

5. Ha, C.T., Nguyen, T.T., Bui, L.T. and Wang, R., 2017, April. An Online Packing Heuristic for the Three-Dimensional Container Loading Problem in Dynamic Environments and the Physical Internet. In European Conference on the Applications of Evolutionary Computation (pp. 140-155). Springer, Cham.

The following lists materials (or part) of the publications presented in the thesis:

- Chapter 2: publication [1-5]

- Chapter 3: publication [1]
- Chapter 4: publication [1, 2]
- Chapter 5: publication [3, 4, 5]

# Chapter 2

# Literature review

In the literature, optimisation problems at container seaports are mainly classified into: berth allocation, quay crane (QC) scheduling, stowage planning, stacking and transport optimisation. Detailed overviews of optimisation problems in this field are provided by Meersmans and Dekker [2001], Vis and De Koster [2003], Steenken et al. [2004], Vacca et al. [2007], Stahlbock and Voß [2008]. In order to answer the research questions raised in Chapter 1, the literature review will focus on the relevant work of dynamic BAPs and the variants followed by a review of optimisation-based simulation at ports and container terminals. After reviewing related work, we summarise the identified features and the direction for further improvement.

## 2.1 BAPs

### 2.1.1 Categories and attributes

In Imai et al. [2005], how a vessel is moored at the quay and the relationship between berth and quay were explained. Depending on how the quay can be occupied at each terminal, this paper also categorised BAPs into discrete, continuous and hybrid spatial attributes. 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 the same 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. The draft of vessels was added as another spatial attribute [Bierwirth and Meisel, 2010]. Vessels with a draft exceeding a minimum water depth cannot be berthed arbitrarily.

In terms of the arrival time of vessels, a BAP is also referred to as static or dynamic [Imai et al., 2001]. Static BAPs generally assume that all the vessels have arrived at the port from the beginning of the time horizon. On the other hand, vessels come with certain arrival times in dynamic BAPs. It means that a vessel cannot berth before its arrival time. Other temporal attributes mentioned in Bierwirth and Meisel [2015] are cyclic and stochastic. With cyclic arrival time, vessels visit a terminal at a certain time repeatedly. For example, a set of vessels visit a terminal at a fixed time every week. A weekly schedule can be made for repeated use. Regarding stochastic attributes, arrival times are defined by continuous or discrete distributions. Instead of the arrival time, some problems are restricted by departure times. It means that each vessel has to either leave before a certain time or wait to be served no longer than a certain time period.

### 2.1.2 Optimisation approaches for BAPs

A review of existing optimisation algorithms is important for dealing with real-world problems. In other words, to work on more realistic optimisation problems, an understanding of relevant works on optimisation problems of other researchers is necessary. The discrete, continuous and hybrid BAPs have been proved to be NP-hard [Hansen and Oguz, 2003, Lim, 1998]. Most optimisation methods used to solve dynamic BAPs in existing work can be grouped into three types: exact methods, heuristics, meta-heuristics. Existing detailed surveys can be found in Bierwirth and Meisel [2015, 2010], Kovač [2017].

#### 2.1.2.1 Exact methods

Exact methods provide optimal solutions but are computationally expensive. Main solution strategies used to solve BAPs are *Lagrangian relaxation, branch-and-cut, branch-and-bound*. As an early work, the static BAP (SBAP) was formulated in Imai et al.

[1997] and it was extended as a dynamic BAP (DBAP) firstly in Imai et al. [2001]. An improvement of Imai et al. [2001] was made in Imai et al. [2003] with ship priority considerations. Imai et al. [2014] improved the previous sub-gradient procedure with Lagrangian relaxation and applied the procedure to a berth template problem which finds a set of berth windows within a fixed time horizon. In Buhrkal et al. [2011], several existing mathematical models of discrete dynamic BAPs were reviewed and compared including Imai et al. [2001], Monaco and Sammarra [2007], Christensen and Holst [2008]. Buhrkal et al. [2011] also provided a generalised set partitioning model with the aim of minimising the total service time. In the model from Buhrkal et al. [2011], each column describes a feasible assignment of each vessel occupying a certain berth at a certain time. Then the model looks for a combination of columns that all the assignments are feasible and with a minimum cost. This approach was also applied in Lalla-Ruiz et al. [2016] in order to solve the problem with multiple tides using the commercial solver CPLEX. Qin et al. [2016] discussed the changing water depths over tides and proposed an Integer Programming and a Constraint Programming for both static and dynamic BAPs. Unlike the tidal constraints considered in Lalla-Ruiz et al. [2016] that low tides and high tides change alternately, in Qin et al. [2016] the water depth changes at every specific time.

Sheikholeslami et al. [2014] proposed a mathematical model for a continuous BAP with tidal constraints. The model ensures that the water-level is high enough for the entry and departure of vessels. Authors suggested a statistics-based sample average approximation method to solve the problem with uncertain arrival time in Sheikholeslami and Ilati [2017]. The disruptive effects of tides were taken into consideration as well. Numerical experiments were conducted based on data from a real-world case. In Dadashi et al. [2017], the tidal windows were formulated to restrict vessels with deep drafts to departure only when the water-depth is sufficient.

Umang et al. [2011] aimed to solve hybrid BAPs in bulk ports. Depending on vessel requirements and cargo properties, the handling time is different. The quay is discretised into a number of sections where each vessel can occupy more than one section if needed.

#### 2.1.2.2 Heuristics

In general, heuristics apply rules and criteria to achieve good quality solutions with a modest computing time. Not many heuristics were used for BAPs in the literature. Xu et al. [2012a] proposed a heuristic approach solving BAPs with two tidal windows. The heuristic is deterministic with pre-set rules which always ensures feasible solutions. Vessels are assigned 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. Because it only accommodates two tides and the second tide goes to infinity, it sometimes obtains a solution with a large number of vessels assigned to the second tide when the number of vessels increases.

Two papers proposed heuristics for continuous BAPs. Guan and Cheung [2004] applied a tree search procedure to explore the search space and developed a lower bound to speed up the search. In the experiment, the tree search approach was proved to solve instances with up to 15 vessels. In Wang and Lim [2007], a stochastic beam search algorithm was developed to minimise the cost of vessels. The approach consists of multiple levels and one vessel is allocated in each level. In each level, estimation, selection and expansion processes are applied for the purpose of seeking effective allocations and increasing the diversity. Furthermore, the approach was proved to solve problems with up to 400 vessels in a small amount of time.

#### 2.1.2.3 Meta-heuristics

**Tabu search** Tabu search (TS) is a meta-heuristic for controlling a heuristic not to be trapped in a local optimum. It explores the neighbours of a solution in order to potentially improve it. An essential feature is a tabu list which uses the memory to record information in the exploration process. The types of memory structures in a tabu list include short-time, intermediate-term and long-term. The algorithm uses a tabu list is to guide the exploration by avoiding some less good solutions.

In Cordeau et al. [2005], a discrete BAP with dynamic arrival times and due dates was addressed. It presented a TS method and was tested on a data set from a terminal in the Port of Gioia Tauro, Italy. The algorithm has two procedures for initial solutions. At the beginning, *Random Greedy* initialises a random queue of vessels and each vessel is inserted to the schedule with the minimum cost evaluation. TS explores neighbourhood solutions and stores the best solution. The algorithm then restarts with the *First-come-First-serve* (FCFS) initialisation procedure that sorts vessels in the queue according to their arrival time. The TS was improved in Lalla-Ruiz et al. [2012] by adding a swap move to the local search and an elite set of solutions. The elite set is used by the path relinking algorithm to generate new initial solutions. The objective of the path relinking algorithm is to iteratively bring the starting point closer to the elite solutions. Giallombardo et al. [2010] aimed to maximise the total value of chosen QC profiles and minimise the housekeeping cost generated by transshipment flows between vessels. A TS was proposed to firstly choose QC profiles for vessels and then optimise the berth allocation schedule with a given QC assignment. In order to reduce the cost in each iteration, the profiles are updated relying on mathematical programming.

Zeng et al. [2011] aimed to solve the continuous BAP and QC assignment problem while taking disruption recovery into consideration. A TS combined with a local rescheduling strategy was designed for the problem. In berth reallocation, a time window and a space window are defined for TS looking for a new solution.

A hybrid BAP was tackled in Lee et al. [2012] with the goal of minimising the cost generated by transshipment flows. A Tabu list recorded the positions of pair-wise interchanges following the first-in-first-out rule.

**Variable Neighbourhood Search** Variable Neighbourhood Search (VNS) was designed for the first time in Mladenović and Hansen [1997] for the purpose of solving combinational optimisation and global optimisation problems. The concept of VNS is changing the neighbourhood during the search based on three principles: 1) a local minimum for one neighbourhood structure may not be the local minimum for another structure, 2) a global minimum is always a local optimum, 3) in many cases, local optima are relatively close to each other with respect to several neighbourhood structures. The VNS expands the neighbourhoods of a given optima until a global improvement is found.

A VNS heuristic was proposed by Hansen et al. [2008] aiming to solve a discrete dynamic BAP. It minimised the cost of waiting time and handling time with the constraint that berths start to be available from different times. The local search called Variable

Neighbourhood descent (VND) consists of three neighbourhoods: local insertion, interchange and insertion. VND is applied as the first phase and then in the perturbation phase, two sets of nested neighbourhoods are used in order to improve the performance of the local optimum. The first one is to change the berths and orders of being served of two randomly selected vessels. In the second one, a random vessel is moved to another random berth at the best position. According to the experiments, VNS outperformed other meta-heuristics and it dealt with large-scale test cases of up to 20 berth and 200 vessels.

**Genetic Algorithm** A genetic algorithm (GA) is a nature inspired algorithm. It was invented based on the idea of using the power of evolution to solve optimisation problems. A GA works by evolving a set of individuals towards better solutions.

A GA was applied to a dynamic discrete BAP in Theofanis et al. [2007] in order to minimise the total weighted service time of vessels. Considering that crossover and mutation operators are highly affected by the problem domain, the author conducted experiments with and without crossover. It was proved that the crossover produces a large number of produced infeasible solutions. A branch-and-bound algorithm then is applied to reallocate vessels as an optimisation component. According to the experiments, the optimisation component effectively improved the quality of solutions but it was time consuming.

Golias et al. [2009a] tackled a multi-objective BAP while considering the berth availability and the priority of vessel services. A multi-population multi-selection GA was proposed with the aim of finding a good-quality solution for each objective function. It also maintains the diversity of different solutions in the Pareto Front. In each iteration, the Pareto Front is updated and the parents and elites are selected based on the Pareto Front optimal set.

Another GA was used to solve a multi-objective BAP [Golias and Haralambides, 2011]. It minimises the tardiness and waiting time of vessels and maximises the premium from vessels early departure. The GA was introduced in Golias et al. [2009b] with a two-layer chromosome representation. The first layer states the arrival time of each vessel, and the second layer represents the service order of this vessel. The authors proposed four types of mutations *insert, invert, swap* and *scramble* that are applied to all the population in each iteration. In the progress of evolution, the weight of doing invert and scramble shifts to insert and swap. It ensures that at the beginning stage the algorithm performs more long jumps, and when it comes to small regions, the GA focuses on intensive improvement.

In Golias et al. [2010], a lamda-optimal heuristic was proposed. The optimal solution can be achieved by exchanging lamda instances of the relation between berths and vessels. In order to avoid expensive computational time while the number of lamda increases, the authors proposed a GA which efficiently reduced the runtime of medium to large scale instances. Golias [2011] aimed to solve BAPs with uncertain vessel handling times. A GA was proposed with two objectives: minimising the schedule risk and total service time. The risk measure is the expected total handling time based on the probability of a berth taking a certain handling time to serve a vessel.

Golias et al. [2014] proposed a GA to solve BAPs with uncertain arrival time and handling time of vessels. Two objectives were minimising the average and the range of the total service times for vessels. The GA initialises the chromosome based on FCFS with early start and FCFS with early finish. Then the maximum and minimum values of each solution are calculated for fitness evaluation. In each generation, Pareto front selects non-dominated solutions which meet both objectives the best. In Han et al. [2010], uncertain vessel arrival time and handling time were taken into consideration as well in berth and QC scheduling. A GA framework was applied with the Monte Carlo sampling for the purpose of performance evaluation. Zhou and Kang [2008] tackled the same problem with a GA. Two sub-strings were used to encode each individual. The first sub-string reflects the berth allocated to each vessel and the second sub-string indicates the service order of each vessel in the berth. Before encoding, vessels are sorted by their arrival time. The search space is then reduced by limiting the vessel number for each berth.

Saharidis et al. [2010] proposed a bi-level hierarchical framework in order to solve BAPs with conflicting objectives: maximising the total throughput of the port and maximising the preferential customer satisfaction. A GA based on the k-th best algorithm was designed for the upper level and the solutions are sent to the lower level in the order of solution quality.

In Hu [2015], the daytime preference was considered as a purpose of optimising BAPs. It was formulated as a multi-objective problem, minimising the total delayed workload and the total night workload. The chromosome consists of two parts: a vector to represent the priority (reflecting the daytime preference) of serving vessels, and a vector to represent the relative berthing time. The algorithm considers the vessel with the highest priority in a priority assessment window of a certain length. Then a berth is selected with the goal of maintaining the minimum increment of cost. The crossover and mutation operations are applied separately to priority vectors and time vectors. Due to the lack of existing research on daytime preference, there was no comparison with other work. However, detailed experiments were conducted to assess the performance of each operator. Lee and Wang [2010] integrated the BAP and the QC assignment problem. A typical GA was applied to find the best sequence of proceeding vessels. When choosing a berth for each vessel, an approximate handling time of serving this vessel in each berth is estimated based on the number of ship bays of the vessel and the number of QCs of the berth.

Lalla-Ruiz et al. [2014] applied a GA to a tactical BAP aiming to determine berth positions, berth time and allocations of QCs for incoming vessels. The tactical BAP aims to allocate vessels to their favourite berth positions as vessels are expected to arrive periodically. In Lalla-Ruiz et al. [2014] the chromosomes are represented as vectors where the berthing order is defined by the first part and the QC assignment is held by the second part. The biased random key generates numbers in [0, 1). By doing vector computations on real values, the berthing order is decided. The crossover selects one parent from the elites and another parent from the rest of the population. A greater probability is given to the elite parent so it is more likely the child inherits keys from its elite parent.

In terms of existing work for continuous BAPs, Chang et al. [2010] combined a parallel GA and a heuristic to minimise the deviation between berth locations, the total penalty of delay and the total energy consumption of QCs. The heuristic was used to initialise the population as feasible solutions. The sub-optimal solutions of BAP and QC assignment were obtained by the GA. Rodriguez-Molins et al. [2014] aimed to minimise the service time for BAP and QC assignment. The service time and robustness were defined as two objectives and a GA was proposed to find non-dominated solutions.

Zeng et al. [2017] proposed a GA to minimise the operational cost of resources and the delay cost of vessels of continuous BAPs. The algorithm consists of two levels. Feasible berthing positions are obtained in level 1 and the storage plan is optimized in level 2 based on the berth allocations. The chromosome representation is the distance between the berthing location of each vessel and the beginning of the quay. Infeasible solutions are fixed in level 1 and the offspring after doing genetic operators in level 2 are sent back to level 1 for the next generation.

Ji et al. [2017] modified the traditional NSGA-II to solve a continuous BAP with tidal constraints. A biased search towards the feasible region is suggested in the approach which utilises the superiority of feasible solutions. The algorithm also combines the evolution population and the solutions in archive in order to not miss promising genes in infeasible solutions. In Yu et al. [2018], authors applied a GA to obtain the optimum berthing schedule with QCs. Vessels are assumed to be moored at their desired positions. In each iteration, the conflicts between vessels are identified and adjusted following several rules.

A hybrid BAP with the purpose of minimising the total service time was solved by a GA [Imai et al., 2013],. The chromosome is represented as a string. The crossover exchanges the sub-strings of two individuals and then examines the feasibility of the children. In the decoding procedure, the vessels are assigned following the service order. Two small vessels might be served at a berth simultaneously if their arrival time and handling time meet one of the four proposed conditions.

Nishimura et al. [2001] incorporated the water depths of berths and vessel drafts to a hybrid BAP. Based on the arrival time, the sequence of vessels is divided into a number of sub-problems. The first sub-problem is solved by a GA and the solution is then sent to the following sub-problem, until the final sub-problem is solved. Two types of chromosomes were proposed and the performance was investigated in numerical experiments. Imai et al. [2007] aimed to solve a hybrid BAP with mega-ships only being served at indented berths. A GA was applied to find an optimal order of scheduling vessels and numerical rules were proposed to assign each vessel.

**Particle Swarm Optimisation** Having a number of particles in the search space, Particle Swarm Optimisation (PSO) searches the best solution by moving particles around. The movement of each particle is based on a velocity and the positions of itself and some other particles. The velocity controls the direction and the length of a jump. The idea of PSO is moving particles towards the best known positions.

Ting et al. [2014] developed a PSO dealing with dynamic discrete BAPs. It was tested on the data set [Cordeau et al., 2005] with an objective of minimising the total service time. In the PSO, the problem is treated as a vehicle routing problem that a vehicle route represents a berthing sequence. Each particle is represented as a real number where the integer part describes the berth that the vessel is assigned to and the fractional part describes the service order of the vessel. Boundary situations are handled after every generation to prevent infeasible solution space. After the search of the solution space by PSO, a local search procedure was applied to the best found particle.

**Simulated Annealing** Simulated Annealing (SA) algorithm is inspired by a physical technique of annealing of solids. It explores the space of neighbourhood solutions based on three parameters: initial temperature, cooling rate and temperature length. The temperature is a function of which iteration it is on. In each iteration, the solution has a probability of whether moving to another state. SA allows backward moves to avoid getting stuck at a local optimum.

In Barros et al. [2011], an SA was proposed to deal with a discrete dynamic BAP considering the priority of cargo stock level and tidal windows. The problem was commonly observed in the maritime industrial port complex located in So Lus. Solutions are formed as a sequence of ships which describes the service order. The basic idea of this approach is to try to exchange a high-cost vessel with a low-cost vessel in the permutation. According to the experiments, it showed good-quality solutions in most of the instances compared to the CPLEX solver. However, the scale of the test instances is at most 3 berths and 30 vessels.

An SA was combined with a clustering search in de Oliveira et al. [2012]. They generated solutions with SA, then used a clustering process to group solutions at each temperature and applied a local search. The structure in SA consists of *reorder ships, reallocate ships and change ships.* Randomness is taken into account in movements of SA to retain diversity. The concept of clustering is to identify clusters with promising regions. Solutions are defined as in the same group by measuring their distance. Lin and Ting [2014] represented the chromosome as a service order in each berth connecting by a zero. An SA was applied after the FCFS based initialisation. It also allows a restart strategy if the current best solution is not improved in a number of consecutive temperature decreases.

Regarding continuous BAPs, Kim and Moon [2003] minimised the total cost of departure delay and the cost resulting from the non-optimal berth of each vessel. Authors applied an SA to deal with the continuous problem. In the encoding and decoding procedure, the stability of the solution is checked to ensure the local optimum. Each iteration explores the neighbourhood solutions where vessels are exchanged in pairs.

In Xu et al. [2012b], authors proposed an SA to solve robust continuous BAPs while the vessel arrival delay and handling time were considered. In order to reduce the search space in the first place, a set of lowest-left tight solutions are constructed based on a fixed buffer time and the given vessel sequence. In the SA, the constructed solutions are divided into subsets and the optimal solution of each subset is found by applying the branch-and-bound method.

Yuping et al. [2017] proposed an SA to minimise the vessel penalty cost, total waiting time and QC assignment. The SA was designed based on the fairness maximisation of this multi-objective problem. A neighbourhood searching algorithm was used to generate new solutions in each iteration. The algorithm searches for local optima while changing the searching structure systematically in order to explore the new searching range.

In terms of hybrid BAPs, Moorthy and Teo [2007] investigated the robust tactical berthing plan by modelling it as a rectangle packing problem. The chromosomes were encoded as sequence pairs. Each pair contains two permutations of a template. A standard SA was then applied to explore the search space of all sequence pairs. Lin and Ting [2014] developed an SA for the hybrid BAP where the quay was segmented into berths and they can be combined to serve one vessel. The chromosome is represented as a sequence of service order of vessels. When assigning each vessel, the algorithm firstly obtains the earliest time the vessel can be moored. The position is then decided as close to the beginning of the quay as possible.

**Other meta-heuristics** There are some other meta-heuristics applied to BAPs not classified in above sections. Relevant work including less popular meta-heuristics in this field, meta-heuristics with exact techniques embedded and other methods related to meta-heuristics are reviewed in this section.

Under the same assumption as Buhrkal et al. [2011], an Adaptive Large Neighbourhood Search (ALNS) meta-heuristic was proposed in Mauri et al. [2016] which is capable of solving both discrete and continuous BAPs. ALNS is an extension of large neighbourhood search and it consists of destroy and repair operators. At each iteration, a destroy operator and a repair operator are selected and applied to the current solution based on a given probability. Destroy operators are chosen from *First shaw removal, Second shaw removal, Worst removal and Random removal*. Repair operators includes *Regret-k insertion and Deep greedy insertion*. The probability of selecting each operator is updated at every iteration according to the performance. In Lee and Jin [2013], the authors proposed a Memetic algorithm (MA) to solve a BAP for cyclically visiting feeders and allocating the storage yard space to the transshipment flows between mother vessels and feeders. MAs are extensions of traditional evolutionary algorithms. An MA generally includes a population-based approach with separate individual learning methods. A GA and TS were combined in Lee and Jin [2013] as an MA. The GA searches the solution space genetically and TS is applied to each offspring for further optimisation.

Lalla-Ruiz et al. [2015] presented a POPMUSIC (partial optimisation meta-heuristics under special intensification conditions) framework. The author pointed out the limitation in Xu et al. [2012a] and limited the second tidal window to the same length in their problem. The POPMUSIC framework was firstly proposed by Taillard and Voß [2002] and it starts from a random permutation, which is to be improved by solving a mathematical formulation. In Lalla-Ruiz et al. [2015], all the associated vessels in a berthing time interval formulate a sub-problem and it is solved by the CPLEX solver. Lalla-Ruiz and Voß [2016] extended the work by adding two common ways of initialisation, *Random-greedy* and FCFS, which were both proposed by Cordeau et al. [2005].

More recently, a dynamic programming-based meta-heuristic was proposed by Nishi et al. [2017] with a goal of reducing the computational time in Lalla-Ruiz and Voß [2016]. It derives lower bounds and upper bounds by a Lagrangian relaxation and a dynasearch algorithm respectively. A machine learning-based system was also applied in a bulk BAP in De León et al. [2017]. The system is trained by running a set of available heuristics and meta-heuristics and then it provides its best solution for each problem.

The Greedy randomized adaptive search procedure(GRASP) has been used in many continuous BAPs. GRASP is a multi-start meta-heuristic which consists of two phases. In each iteration, the construction phase makes complete solutions and then the local search looks for the local optimum of each solution. Two GRASPs were proposed in Lee et al. [2010] in order to minimise the total weighted turnaround time of vessels. The first GRASP initialises solutions based on the first-come-first-pack rule which is not the priority in the second GRASP. Swapping vessels and an A-start like tree search were applied as local search in both versions of GRASP. The first GRASP only swaps two adjacent vessels because it aims to follow the first-come-first-pack idea while the other GRASP can swap any two vessels. A continuous BAP with QCs was also solved in Salido et al. [2011] with GRASP. The initialisation strictly restricts that a vessel with an earlier arrival time has to be moored before later vessels. Following the order of vessel arrival times, a branch-and-bound method explores the complete search space for vessel insertions. This approach was integrated in a decision support system in Salido et al. [2012]. To avoid expensive computational time, the local search was not included in their algorithm.

Yang et al. [2012] applied an evolutionary algorithm with nested loop for BAPs with QC assignment. The authors focused on the interactions between two sub-problems and the feedback of them. With presumed vessel handling time, GAs were used as two inner loops for two sub-problems respectively. The output handling time is calculated by an outer loop and the value is returned to the inner loop as the input in the next iteration.

Ma et al. [2017] proposed a Guided Neighbourhood Search (GNS) to tackle a hybrid BAP. In this paper, GNS decides critical elements by objective values which are updated in each iteration. Given a priority sequence of vessels as the chromosome, vessel movements to promote or lighten the priority are used as operators. The algorithm applies different movements to critical elements based on multiple suggested criteria. Xiang et al. [2017] introduced an Adaptive Grey Wolf Optimiser (AGWO) to solve a bi-objective robust problem. Grey wolf optimiser mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. The population is divided into four groups for operations including hunting, searching for prey, encircling prey and attacking prey. In this paper, the chromosome includes information of the berthing position and the time to berth of a vessel. A preliminary experiment was conducted to determine the size of different groups of wolves.

Cheong et al. [2010] applied an evolutionary algorithm to deal with a hybrid BAP with multiple objectives. It aimed to minimise the makespan, the total waiting time of vessels and the total number of crossings between vessels. Two different decoding schemes were proposed and investigated. One is assigning a vessel to the time no earlier than the last one. The other scheme allows a vessel to be served at the earliest time if there is enough space. Extensive experiments were conducted to examine the quality of each operator for different objectives. Umang et al. [2013] proposed a Squeaky Wheel Optimisation (SWO) to tackle hybrid BAPs. SWO constructs solutions in a greedy way and then specifically pays attention to the trouble spots. New solutions are generated based on elements with priorities that are most likely to be improved. In Umang et al. [2013], the solutions are initialised in the FCFS base. At the end of each iteration, the priority order of vessels is generated according to the individual contribution to the overall service time. Therefore, the new rank of vessels in the next iteration is the order of the current performance of each vessel. The aim is to move individuals in priority order to positions with minimised total waiting time and handling time.

### 2.1.3 Discussions on optimisation approaches of BAPs

The literature review above shows that meta-heuristics are the most preferred methods to solve the BAP in a majority of existing research. This agrees with the finding from Bierwirth and Meisel [2015] that 40% of reviewed work used GA or evolutionary algorithms and another 36% used other meta-heuristics. 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.

According to the review, the performance measures of BAP models mostly focus on the total stay time of vessels at the port. The stay time depends on constraints such as berth availability, loading and unloading cargoes and the availability of other resources like QCs. The stay time is reflected to different objective functions in literature:

**Minimising the total waiting time**

The time each vessel spends between arrival and being served.

**Minimising the total completion time**

The time each vessel spends between arrival and departure. It is usually the summation of the waiting time and the handling time. The handling time of each vessel sometimes can vary.

**Minimising the total cost of waiting**

In general, the cost can be 1) a penalty for exceeding free stay time, 2) a time-unit cost of staying at the port.

BAPs have received a lot of attention and a large number of valuable publications have appeared over the past decade. We can see a significant improvement in academic research in solving BAPs. Still, there are some characteristics that have not been well-studied yet. Some weaknesses in dealing with real-world problems we have identified are as follows.

**Multi-tidal constraints** At seaports, changing tides happen every day. They affect the availability of berths because the water level of a berth is changed when the tide changes. Over the attributes and problem constraints we have reviewed, the importance and necessity of tidal constraints are highlighted in a recent survey paper [Bierwirth and Meisel, 2015] but there are very few publications actually dealing with it. Two tidal windows were considered in Lalla-Ruiz et al. [2015] and Xu et al. [2012a]. Multi-tidal constraints were dealt with by the CPLEX model in Lalla-Ruiz et al. [2016] and Qin et al. [2016] modelled the problem while considering vessel drafts to match time-varying water depth. To the best of our knowledge, there is no existing meta-heuristic used in dealing with multi-tidal constraints. We observe that existing exact methods can be restricted when conducting large-scale experiments. The computational time is usually large due to the limitation of commercial solvers. Sometimes a quick response to a new schedule is essential in practice. There is no existing work that guarantees a feasible solution and a quick turnaround time.

**Benchmarks** As far as we know, the data set from Cordeau et al. [2005] is one of the most popular test instances used in existing experiments. In Cordeau et al. [2005], the author conducted a statistical analysis on the industrial data from Gioia Tauro in order to generate test instances to simulate real-world scenarios. Benefitting from the approach of generating test instances provided in this paper, for some other dynamic BAPs with more variables required, authors were able to generate their own data based on the concept in Cordeau et al. [2005]. However, there is no benchmark that covers different types of BAPs. The lack of such benchmarks brings up the difficulty in comparing different problem solutions.

**Large scale dataset** As the globalisation of maritime transport develops, ports have become busier than ever before. As mentioned above, the scale of Cordeau et al. [2005] is limited and the largest scale instance we have reviewed in existing work is 200 vessels in Hansen et al. [2008]. In reality, there are many ports receiving more than 200 vessels per planning period. It indicates a need of large scale test datasets.

**Performance measures** As summarised above, most BAPs focus on evaluating stay time of vessels at the port and related costs. A few BAP models aim to reduce the utilisation of resources like cranes and berths. Moreover, operational performance indicators like fraction of time berthed vessels worked, berth throughput have been suggested in UNCTAD, 1976 and de Langen et al. [2007]. There is no existing work comparing the performance of their algorithms if two BAPs have different objective functions.

## 2.2 Simulation at ports

### 2.2.1 Introduction of simulation

Modelling is a way of experimenting solutions in order to solve real-world problems. Sometimes conducting experiments on real systems is expensive or impossible. Simulation provides us a way to mimic the reality and how a model will change the system over time. Simulation has been applied to a wide range of problems such as supply chain, transportation, computer hardware and automotive control system etc. [Borshchev and Filippov, 2004].

In Angeloudis and Bell [2011], simulation models were categorised based on three criteria.

**Static or dynamic**

Static simulations focus on time-independent problems. In contrast, dynamic simulations consider how a system changes over time.

**Microscopic or macroscopic**

This term describes the fidelity of the simulation model. Microscopic models focus on details of a system. The environment and inputs are set specifically. Macroscopic models aim at the strategic level of a system with high abstraction.

**Continuous or discrete**

Continuous models are able to describe any specific time of the system. They are commonly used in physical sciences and finance. With dynamic models, the states of a system are split and changed from one to another. This type of simulation is usually used in logistics and manufacturing. Moreover, the discrete simulation can be identified as two types: *discrete time* and *discrete event*. The discrete time simulation consists of a number of time steps. The more time steps in the model, the higher the accuracy of the simulation. Because after each time step the model needs to update the states of all the objects, the possible high complexity is the drawback of this type of simulation. On the other hand, the discrete event simulation is controlled by an event manager with a list of possible events. The simulation reacts only if an event happens.

### 2.2.2 Applications of simulation in port operations

The simulation modelling of port operations has been progressed gradually with the growth of information technology and the increasing demand of port operations over the past 20 years. According to Dragović et al. [2017], there is a constant increase of relevant publications from 1995 to 2015, especially after 2011, a remarkable number of simulation models of port operations have been published.

The survey paper [Dragović et al., 2017] summarised the related work and reviewed it in detail. Among various simulation software/libraries used in existing work, Arena is the most-used software. In addition, discrete event simulation is one of the most popular simulation techniques in modelling port operations. In this section, we will focus on reviewing existing work related to optimisation integrated to simulation.

Arango et al. [2011] simulated a BAP in Seville inland port with Arena software and optimised the problem by proposing a GA. The algorithm aims to minimise the total service time of vessels. A number of simulations were executed and the statistical results were compared with the original system. Legato et al. [2014] integrated a heuristic to the simulation model in order to solve tactical BAPs. A discrete event simulation model was developed by a Monte Carlo simulator while taking into account uncertainties in unloading cargoes. The authors used the simulation to evaluate the tactical berth template at the operational level. Then they compared the solution with existing methods. He et al. [2015b] tackled a yard crane scheduling problem aiming to optimise the energy consumption and service efficiency. An approximate approach was designed by combining GA,PSO and the simulation model. Feasibility of solutions was checked and infeasible genes were repaired by running simulation. However, the simulator name is not mentioned in the paper. A following work based on He et al. [2015b] integrated QC scheduling and internal truck in He et al. [2015c].

Legato et al. [2010] proposed a discrete event simulation model in order to deal with QC scheduling problems. An SA was introduced to search for the optimal solution. The performance was tested by simulating the real-world scenario in Gioia Tauro, Italy. QC scheduling and storage planning were also simulated in Zeng et al. [2015]. A GA was integrated to the simulation model. Once an optimal solution is obtained by the GA, the performance is evaluated by the simulation while taking into account the uncertainty of loading and unloading time. Ilati et al. [2014] aimed to simulate the resource allocation including berth allocation, QC assignment and tugboat assignment. An evolutionary path-relinking algorithm was introduced to optimise the assignments. The simulation model was developed using commercial software Enterprise Dynamics and tested on Rajaee port in Iran.

Cordeau et al. [2015] focused on simulating the container transshipment while reducing congestions and speeding up the loading and unloading process. The simulation also embedded a local search procedure in order to improve the solution of vehicle scheduling. Li and Wang [2009] simulated the whole operation system of container terminals and optimised the truck distribution. A parallel computing technique was applied to the simulation which effectively saved the computational time. Clausen and Kaffka [2016] aimed to optimise the crane control problem by defining a priority number for each container. The container with the highest priority is sent to the next crane. The efficiency of the overall operational system was tested by the simulation software ContSim with existing data and newly generated data.

In Zehendner et al. [2015], straddle carrier allocation was optimised and simulated with the goal of optimising the overall delay. The author introduced a mixed integer programming model to represent the network flow strategy. The simulation model was developed in Arena and experiments were conducted on the data from the Grand Port Maritime de Marseille. The simulation was applied to validate the solution and then test with stochastic settings. With the goal of optimising the problem of sharing internal trucks among multiple container terminals, in He et al. [2013] the simulation was used to check whether the solution is feasible while the proposed GA is responsible for exploring the search space. Another GA combining with PSO as a local search was developed by He et al. [2015a] to optimise the supply chain network. Similar to He et al. [2013], the simulation model was proposed to evaluate the model and repair infeasible solutions.

### 2.2.3 Discussion on simulation integrated with optimisation

The review above discusses various simulation models developed with different software, mostly commercial software. Researchers aimed to solve different problems at ports and container terminals with different objectives. Numeric work was tested with the data from Gioia Tauro, Italy. As far as we know, the existing work of optimisation integrated to simulation only focused on their own approach. None of them had a fair comparison of optimisation algorithms by either integrating other algorithms in the same simulation model or providing a tool for potential integration and comparison. It leads to an important gap: the lack of a general work/platform that allows compassion of different algorithms on the simulated environment. We will try to close the gap by proposing a simulation framework for ports in Chapter 5. This framework will be able to integrate with optimisation algorithms. It also generates test cases for comparison of different algorithms. The framework will also be user-friendly, so that even researchers without programming experience should be able to develop simulation models and evaluate their optimisation algorithms.

## 2.3 Summary

In this chapter, we firstly reviewed the optimisation algorithms of BAPs. Shortcomings of existing approaches have been discussed in Section 2.1.3. Then the simulation integrated optimisation was reviewed and summarised in Section 2.2. In the next two chapters, we will focus on solving BAPs with tidal constraints by proposing new meta-heuristics. This is followed by developing a new framework in Chapter 5 in order to fulfil the need of simulation and optimisation in expanding ports.

Chapter 3

# Solving berth allocation problems with multi-tidal windows using Levy flight

## 3.1 Introduction

In the previous chapter, we found that BAPs have been defined and modelled in academic research with a goal of fitting real-world scenarios. Shortcomings which have not been well-studied in the literature have been summarised. One of the important issues is dealing with multiple tides. Tides are the alternates and falls of the water level of oceans, seas and bays, etc. They are caused by the attraction of the moon and the sun (Fig. 3.1). The gravity of the moon and the sun pulls water away from the earth. Different regions of the earth experience different tidal regimes. Tides occur in a predictable pattern. Around the UK, there are mostly two high tides and two low tides each day. Some other parts may have one high tide and one low tide per day. The different in height between high tide and low tide is called the tidal range (Fig. 3.2).

In reality, 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. For example, Port of Liverpool and Port of Xiamen usually experience large tidal ranges. In Port of Liverpool,

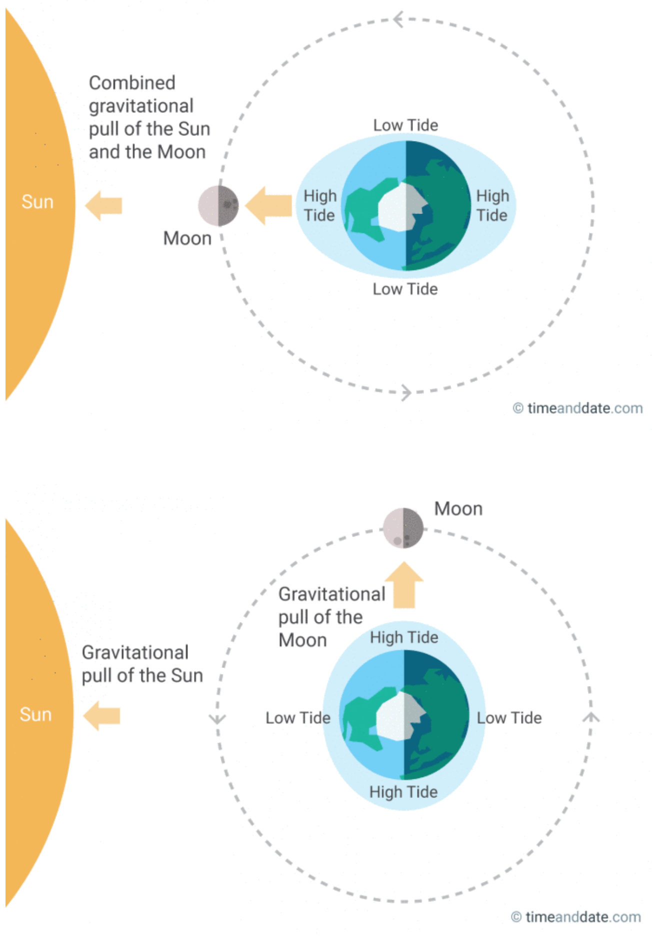


FIGURE 3.1: Tides are influenced by the sun and the moon [timeanddate.com, 2018].

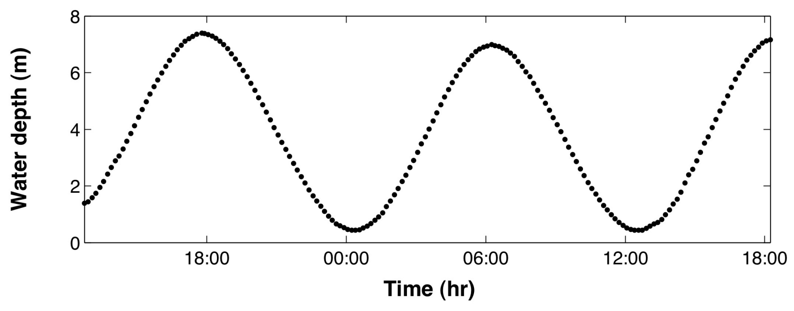


FIGURE 3.2: An example of the tidal range.

there are four (two high and two low) tides per day. The difference in height can go up to 10 metres. The low tide can be less than 1 metre and the height sometimes increases to 10 metres at high tide. It happens similarly in Port of Xiamen in China with a tidal range of more than 5 metres. This kind of ports with a large tidal difference every day, can cause a potential problem of serving big cargo ships due to the insufficient water level. Therefore, considering tidal constraints is essential to allocating incoming vessels to feasible berthing positions.

Different types of BAPs have been reviewed in the last chapter. According to the case study of our industrial partner in Section 1.2, the berths are isolated with their own quays. The traditional continuous BAP would not be realistic since the berth positions are defined as arbitrary and there is only one quay. In discrete BAPs, the length of vessels is not considered and the quay is pre-split to a number of berths. Hybrid BAPs allow a quay to accommodate a long vessel or more than one vessel if the space is enough.

To illustrate the difference between discrete BAPs and hybrid BAPs, an example is displayed in Fig. 3.3. If the problem is discrete, there is Berth 1, Berth 2a and Berth 2b while considering it as hybrid, it consists of Berth 1 and Berth 2. It is noticeable that a hybrid BAP can be also treated as a discrete problem by considering Berth 2 as an additional berth for the discrete BAP. Therefore, the simplified hybrid BAP becomes a discrete BAP including Berth 1, Berth 2a, Berth 2b and Berth 2, with a constraint to restrict the occupancy of Berth 2a, Berth 2b and Berth 2. For example, occupying Berth 2a makes Berth 2 unavailable. In this way, a discrete BAP becomes more suitable for modelling realistic problems. The complexity of the problem can also be reduced by not taking into account the length of each vessel. In this thesis, the BAP will be treated as a discrete problem.

In this chapter, we focus on answering the third research question in Section 1.4 in terms of effectively solving the problem with tidal constraints. The aims of tackling the tidal constraints in discrete dynamic BAPs are: (i) minimising the total weighted service duration of vessels, where the weight is considered as the priority of vessels; (ii) scheduling available berths for the arriving vessels taking into account a multi-tidal planning horizon.

Among existing approaches, the latest approximate algorithm [Xu et al., 2012a] has shown decent results on BAPs with two-tide constraints. Because it only accommodates

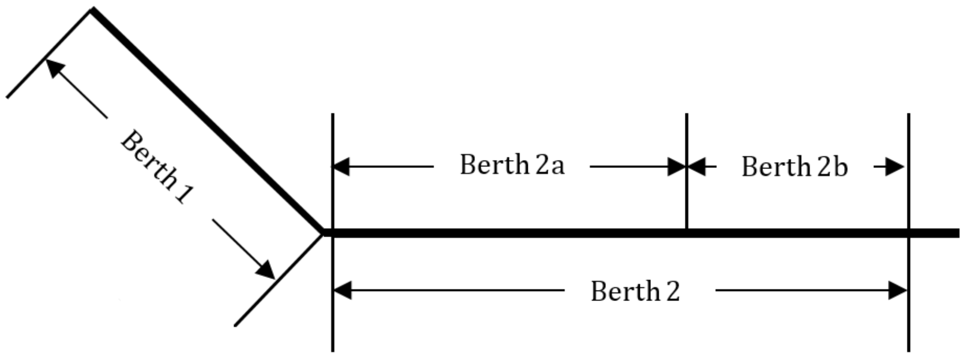


FIGURE 3.3: An example of the layout of a discrete BAP and a hybrid BAP.

two tides, it sometimes obtains infeasible solutions when the number of vessels increases. The approach introduced in Xu et al. [2012a] 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 Xu et al. [2012a] to improve the only solution. In this chapter the new algorithms will deal with multiple tidal windows whereas Xu et al. [2012a] only deals with two tides. Meta-heuristics benefit from the randomness to diversify the solution while Xu et al. [2012a] is deterministic. Another state-of-the-art exact method [Lalla-Ruiz et al., 2016] used a commercial solver to ensure feasible solutions but the capability of dealing with large-scale problems is limited and the computation time is too long to obtain good solutions.

A Levy flight based meta-heuristic is proposed in this chapter. We study how this meta-heuristic performs on BAPs with tidal constraints and compare it with the state-of-the-art exact technique using a commercial solver and other approximate methods. The main contributions of the algorithm are summarised as follows. Firstly, the algorithm provides competitive berth allocation schedules compared to the state-of-the-art exact and approximate methods. Secondly, it is the best algorithm so far that can always achieve 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 chapter is explained here. We describe the problem in mathematical formulations with all necessary constraints in Section 3.2. Then a single-point meta-heuristic Levy flight based algorithm is proposed in Section 3.3 followed by a study of the sensitivity of the proposed algorithm and comparing with the state-of-the-art exact method and heuristic. Highlights of the performance of the algorithm is briefed in Section 3.4.

## 3.2 Problem description

In this section, the BAP with multiple tides is modelled as a discrete problem. The problem is described by introducing assumptions, notations and the mathematical formulation of the objective function as follows.

### 3.2.1 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

When scheduling vessels, many factors need to be considered in practical situations. As mentioned in Section 1.2, tidal effects can be significant for some terminals. The water depth at high tides may be required for some vessels going through certain sections. For simplicity, some practical factors are not considered in this problem but aimed to be tackled in future work. For example, the processing time of loading/unloading a vessel may vary which depends on the resources allocated to the vessel. In other situations, a breakdown of machines can cause delay of the serving process while a vessel normally has to finish the job in a pre-agreed time window. These practical factors are further discussed in Section 3.2.5.

### 3.2.2 Notations

$m$**:** The total number of berths

$n$**:** The total number of vessels

$t\_arr_j$**:** The arrival time of vessel j

$t\_wait_j$**:** The waiting time of vessel j which equals to $t\_start_j$-$t\_arr_j$

$t\_proc_j$**:** The processing time of vessel j

$w_j$**:** The priority of vessel j. A time-scaled cost is generated after vessel j arrives at the terminal.

$TF$**:** The 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

### 3.2.3 Mathematical model

Due to the strong impact of changing tides in practice, it is considered as a restriction to vessels in this problem. In our problem setting, high tides and low tides happen alternately. According to Section 3.2.2, $TF$ 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 draughts of vessels, not all the berths are available for a vessel at all times. 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 $H_j$ is defined 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 shown in Table 3.1, 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 number of berths available at high tide must be no less than that at low tide.

In multi-user terminals, incoming vessels are not always assigned to a specific berth position. The incoming vessels are normally held at the port horizon and waiting for further instructions from the operator. In practice, the operator needs to send tug boats to direct vessels which are going to be berthed. The decisions are based on the schedule and their priorities. The priority policies varies in different ports [Kontovas and Psaraftis, 2011]. In general, large shipping companies may have agreements regarding the priority. In Japan, they give priorities to large vessels because they cause more congestions [Imai et al., 2003]. On the other hand, small vessels may be given service priority when the port is busy, such as at Dalian container terminal.

In literature, Guan et al. [2002] treated the vessel size as a part of the weight with an objective of minimising the total weighted completion time. A large size results a small weight. Guan and Cheung [2004] defined the weight to represent the importance of each vessel and minimised the total flow time. Saharidis et al. [2010] employed weights to distinguish the preference of vessels while considering the customer satisfaction over the ports total throughput. Hansen et al. [2008] minimised the total cost including earliness and lateness costs, and the weighted handling time.

In our model, the service priority of vessels is represented in a form of cost rate $w_j$. Since ships do not have equal importance, a weighted sum of the vessel service times may better reflect the management practice. A vessel with a high priority, means that if it costs more than a vessel with lower priority while waiting. To minimise the cost without the weight priority, set $w_j$ of all vessels to 1.

The objective function (3.1) minimises the total weighted 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.

$$\min \sum_{j=1}^{n} w_j(t\_wait_j + t\_proc_j) \tag{3.1}$$

s.t.

$$\sum_{i}^{m} x_{ij} = 1 \quad \forall j \in n \tag{3.2}$$

$$t\_start_j \geq t\_arr_j \quad \forall j \in n \tag{3.3}$$

$$t\_start_{j'} \geq t\_start_j + t\_proc_j - m(1 - I_{ijj'}) \qquad \forall j, j' \in n, j \neq j', \forall i \in m \tag{3.4}$$

$$I_{ijj'} + I_{ij'j} \leq \frac{1}{2}(x_{ij} + x_{ij'}) \qquad \forall j, j' \in n, j < j', \forall i \in m \tag{3.5}$$

$$I_{ijj'} + I_{ij'j} \geq x_{ij} + x_{ij'} - 1 \qquad \forall j, j' \in n, j < j', \forall i \in m \tag{3.6}$$

$$x_{ij} = 0 \quad \forall j \in n, i = 1, 2, ..., max(L_j, H_j) - 1 \tag{3.7}$$

$$x_{ij} \in \{0, 1\} \quad \forall j \in n, \forall i \in m \tag{3.8}$$

$$I_{ijj'} \in \{0, 1\} \quad \forall j, j' \in n, \text{s.t.} j \neq j', \forall i \in m \tag{3.9}$$

Constraint (3.2) ensures each vessel is assigned to only one berth. Constraint (3.3) requires that the vessel can only be served after it arrives. Constraint (3.4) 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 $t\_start_j + t\_proc_j$.

Constraints (3.5) and (3.6) 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 (3.7) restricts vessel $j$ to be only assigned to a berth always available to it. Constraints (3.8) and (3.9) enforce $x_{ij}$ and $I_{ijj'}$ to be binary.

Table 3.1: 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 |

### 3.2.4 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 (3.1), 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.

### 3.2.5 Potential extensions for practical uses

In order to determine a berth schedule, there are more factors in reality that we need to consider than the above problem assumption. For simplicity, some practical factors are not considered in this problem. However, with the proposed algorithms in following sections, these features can be either added as constraints or further integrated into our algorithms.

#### 3.2.5.1 Time window

In practical scenarios, vessels may have service deadline (the start or finish time of service) in the form of time windows. Due to the agreements between shipping companies and the port, the time windows determine that each vessel has to depart over a period of time after arrival. In some cases, time windows are soft which can be relaxed with a penalty cost. Furthermore, berth windows are also considered depending on the terminal planning. A berth window guarantees a service which meets certain performance standard. Numerous ports are closed at night and over weekends [Christiansen et al., 2007]. This feature becomes important especially when planning for more than a week.

Golias et al. [2007] formulated BAPs considering the departure of vessels within time windows, early and late departures. The BAP was modelled in both Cordeau et al. [2005], Ting et al. [2014] as a vehicle routing problem where each vessel is treated as a customer and has to be served within a period of time. Berths as depots, are restricted by a time window of availability. The due time for departure of each vessel in Park and Kim [2005], Legato et al. [2008] is pre-set and a penalty cost is incurred if the departure is later than the committed time window. In Hendriks et al. [2008], the arrival time window was considered based on an agreement between the operator and shipping lines. Only if the vessel arrives within a certain time window, the port guarantees a maximal time to serve the vessel. Otherwise, the port will not be penalised by any delay of service. Lalla-Ruiz et al. [2016] considered both the available period of time of berths and the maximum departure time of vessels. Authors also pointed that the computational time of the model in is reduced by defining them as hard constraints.

#### 3.2.5.2 Stochastic processing time

Bierwirth and Meisel [2010] summarised different definitions of processing time in literature were:

1. They are known as fixed
2. They depend on the vessels berthing positions
3. They depend on the QC assignments

4. They depend on the QC scheduling
5. They depend on combinations of above.

The real-world processing time of loading/unloading a vessel may vary which depends on the resources allocated to the vessel. In the processing of loading/unloading, QCs are one of the main resources. When several vessels berth simultaneously, the plan of QC assignment restricts the speed of handling each vessel. At least a certain number of QCs are required to serve each vessel. Sometimes the QC capacity and the operation cost rate of QCs are considered as well. In other words, the realistic processing time of each vessel, is influenced by the total number of QCs and the QC assignments.

Chang et al. [2010], Yang et al. [2012] combined BAPs and QC assignments and the processing time was estimated by the required workload and the number of QCs assigned. It was estimated in the same way in Zhou and Kang [2008] based on stochastic probability. Instead of workload, the number of required movements was given in Rodriguez-Molins et al. [2014] for the estimation of processing time. The processing time for BAPs can also be obtained from a QC schedule. The QC scheduling and the BAP were dealt simultaneously [Han et al., 2010, Lee and Wang, 2010, Zeng et al., 2011]. The necessary time of handling each vessel is decided by the number and schedule of QCs. Zeng et al. [2011] also took the disruption of operations into consideration. During operations, severe weather conditions, breakdown of equipment or other unforeseen events could cause disruptions in reality. In Zeng et al. [2011], the disruption was imposed to certain vessels with a time of delay.

## 3.3 Levy flight for BAPs with multi-tidal windows

A single-solution based meta-heuristic optimisation termed LF-BAP is introduced 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 [Tran et al., 2004]. 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 the Levy flight distribution. In BAPs, 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. In the process of LF-BAP only one individual is used. 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 are 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 3.3.1.1) and then updates the current solution by adapting a Levy flight random walk (Section 3.3.1.2). Our decode procedure (Section 3.3.1.3) 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.

**Algorithm 1** The process of LF-BAP

```
current_solution := Initialisation(); //encoding (Section 3.3.1.1)
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 (Eq. 3.11)
For (int i = 0; i < N, i++ ) do:
    randomly pick two positions in current solution
    switch the contents of them
End
```

### 3.3.1 First phase

#### 3.3.1.1 Encoding

To be able to solve the BAP using a meta-heuristic like LF-BAP, it is necessary to find a method to encode all information of a solution into a data structure. In this section, the following encoding structure is proposed: 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 in which 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 3.2. 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 is ordered by their arrival time. The vessel with the earliest arrival time has the highest priority in the order.

TABLE 3.2: Example of encoding a solution given priorities of vessels

| Vessel j | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Priority | 2 | 4 | 1 | 3 | 5 |

$\downarrow\downarrow$

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Vessel allocation order | 3 | 1 | 4 | 2 | 5 |

#### 3.3.1.2 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 [Meng et al., 2016] and Cuckoo search algorithm [Yang and Deb, 2009]. 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 [Terdik and Gyires, 2009]. Levy flight was combined with artificial potential field method in order to perform an efficient searching algorithm for multi-robot applications [Sutantyo et al., 2010]. As mentioned above, lately Levy flight has been effectively applied to optimisation problems with a large search space as it significantly increases the diversity of the chromosome and avoids to be trapped in local optima [Ali, 2015, Viswanathan et al., 2008]. 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 Tran et al. [2004]. The distribution below (3.10) from Tran et al. [2004] is a normalisation of the random walking length distributed in Gutowski [2001]:

$$P(l) = \frac{\beta}{l_0(1+\frac{l}{l_0})^{1+\beta}} \tag{3.10}$$

where $l$ is noted as the step length.

Moreover, Tran et al. [2004] also provided a formulation (3.11) 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 $\beta$ increases.

$$l = l_0(\frac{1}{U^{1/\beta}} - 1) \tag{3.11}$$

In general, Levy flight technique is designed for continuous optimisation problems. To apply LF-BAP to the combinatorial domain of the BAP, a randomly generated step length is rounded to an integer number. This number represents the number of swaps of two random positions in vessel allocation order (Table 3.2). 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.

**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 3.2, $L_j$ represents the set of available berths for vessel j is $\{L_j, L_j+1, ..., m\}$ at low tide.
Let $H$ be a set of available berths for n vessels at high tide $H := \{H_1, H_2, ..., H_n\}$. //$H_j$ represents the set of available berths for vessel j is $\{H_j, H_j+1, ..., m\}$ at high tide.
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
```

#### 3.3.1.3 Decoding

To decode the information from the data structure to a berth allocation solution, the representation in Xu et al. [2012a] is modified. The idea of allocating a vessel to an available berth with the minimum increment of the objective value in Xu et al. [2012a] is used in our decoding process. However, there are only two tides considered in Xu et al. [2012a]. 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 Lalla-Ruiz et al. [2016]. There are only two tides in Xu et al. [2012a] in which the second tide goes to infinity. If there is a large number of vessels arriving, and the first tide can only accommodate a small proportion of them, the second tide has to be much longer than usual. It is not suitable for real-world problems.

As there are more than two tides in reality, to avoid infeasible solutions, additional checks 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, it has to be checked 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) it checks 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.

### 3.3.2 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. Xu et al. [2012a] 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 Xu et al. [2012a].

A new local search is proposed 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 Ting et al. [2014] and the third part is newly proposed in this thesis. 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. A vessel is randomly chosen from a certain berth and inserted into 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, one random vessel is selected 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 $b_3$. Select a random vessel currently allocated to $b_3$ and insert it to a random position in $b_2$. Similarly for $b_1$, a random vessel is chosen from $b_3$ to be inserted to $b_1$. If a move like this leads to an improved berth schedule with a smaller cost, the newly created berth schedule will be kept.

### 3.3.3 Computational experiments

In this section, the performance of the proposed LF-BAP is assessed 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. 16GB maximum java heap size is set for each run by CPLEX.

**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
```

In this work, four sets of problem instances are tested. Set I corresponds to instances used in Xu et al. [2012a], Lalla-Ruiz et al. [2016] with a maximum of 8 berths and 24 vessels. In the work of Lalla-Ruiz et al. [2016], 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 NWEUROPE [2008], Europes two biggest ports in Antwerp and Rotterdam already carried over 400 ships per day in 2008. It was also mentioned in ForConstructionPros [2016] (2016) that Port of Rotterdam handles on average 383 vessels per day. Asian ports can be even busier Wikipedia [2017]. The Port of Singapore 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 [Akanksha Gupta, 2013]. However, according to recent ranking tables, many Chinese ports are deemed larger nowadays [Wikipedia, 2017, World Shipping Council, 2015]. Live vessel trackers [Vesseltracker, 2018] also indicate that many Chinese ports, such as Shanghai, Zhoushan, Qingdao, Ningbo, Tianjin, have up to dozens of hundreds of vessels at ports, and a few hundred expected vessels at any 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 [MarineLink, 2015], Tubaran [Ta Kung Pao, 2011], Cartagena [Andrew Mwaniki, 2018], etc. In addition, the problem considered in this chapter 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 the real world.

In order to simulate challenging real-world problems, we generate large-scale data sets III and IV following the instruction in Xu et al. [2012a]. Regarding the busy ports mentioned above, not all of them show a large tidal difference. The instances are generated in different tidal effects (small and big impacts) in order to simulate scenarios of different ports. The arrival time $t\_arr_j$ and the processing time $t\_proc_j$ of each vessel j are generated according to a discrete uniform distribution within {0, 1, ..., TF} and within

{3, 4, ..., 12} respectively. The weight $w_j$ of each vessel is randomly generated according to a discrete uniform distribution within {1, 2, ... , 10}.

The low-water berth index and high-water berth index of vessel j has a probability of 0.5 generating as follows. $L_j$ is randomly generated according to a discrete uniform distribution within {1, 2, ... , m}, and $H_j$ is set equal to max{$L_j$  1, 1} (resp. max{$L_j$ 2, 1}) for the case where the tidal impact is small (resp. big). Otherwise, $H_j$ is randomly generated according to a discrete uniform distribution within {1, 2, ... , m}, and $L_j$ is set equal to min{$H_j$ + 1, m} (resp. min{$H_j$ + 2, 1}) for the case where the tidal impact is small (resp. big).

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 3.3 and a corresponding feasible solution is displayed (Fig. 3.4). We run LF-BAP 50 times for each instance and one execution for CPLEX-BAP and H-BAP because the solution always stays the same.

#### 3.3.3.1 Comparing with an exact method and a heuristic

**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 [Hansen and Oguz, 2003, Lim, 1998], exact methods may need exponential effort for even medium-sized problems. For example, for each vessel $j$, it may need a binary variable $S_{j,b}$ denoting whether vessel $j$ is sent to berth $b$. Assume there are a set of vessels $V$ scheduled to berth $b$, it is also needed 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 [Lalla-Ruiz et al., 2016]

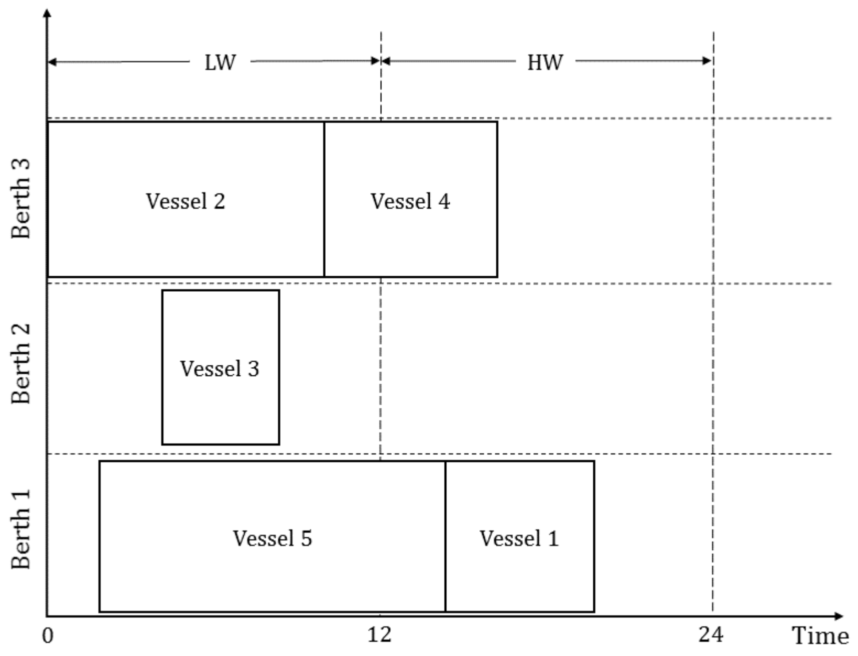


FIGURE 3.4: A feasible solution of the example instance given in Table 3.3

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 3.3: An example instance using notations from Section 3.2.2

| 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 |

**A modified heuristic** In general, heuristics can provide solutions quickly like the greedy heuristic [Xu et al., 2012a]. It prioritises vessels based on the available berths. Vessels with the same number of available berths are grouped together. Vessels in each

Table 3.4: Parameter settings

| Algorithms | Population size | Other parameters |
|---|---|---|
| LF-BAP | 1 | Maximum iterations = 10000; $l_0 = 10$; $\beta = 3$. |
| CPLEX-BAP | Not applicable | Time limitation: 1 hour. 16GB heap size. Other parameters set as default. |

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 Xu et al. [2012a] 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.

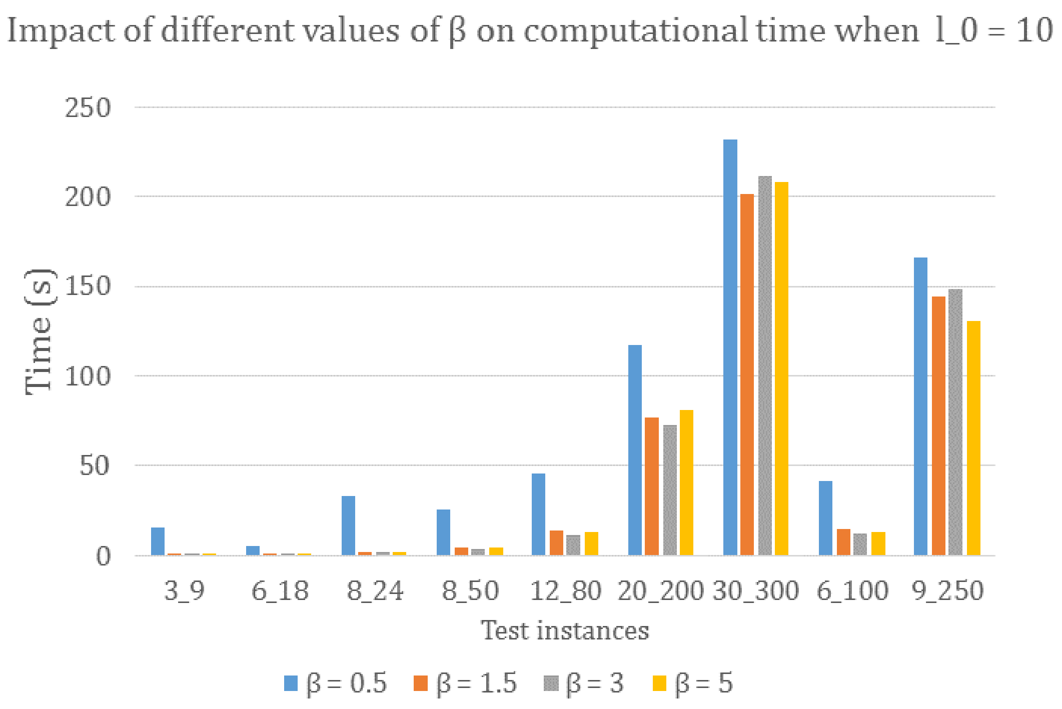
A comparison between LF-BAP and a modified heuristic (H-BAP) from Xu et al. [2012a] would be able to show whether stochastic elements in LF-BAP significantly improve the results. H-BAP repeats the algorithm in Xu et al. [2012a] so that it fits multiple tidal windows. In the original algorithm in Xu et al. [2012a], the second tidal period goes to infinity which is not feasible for real-world problems. With the modification explained in 3.3.1.3, it guarantees feasible solutions to have a fair comparison with our work.

#### 3.3.3.2 Sensitivity analysis of LF-BAP

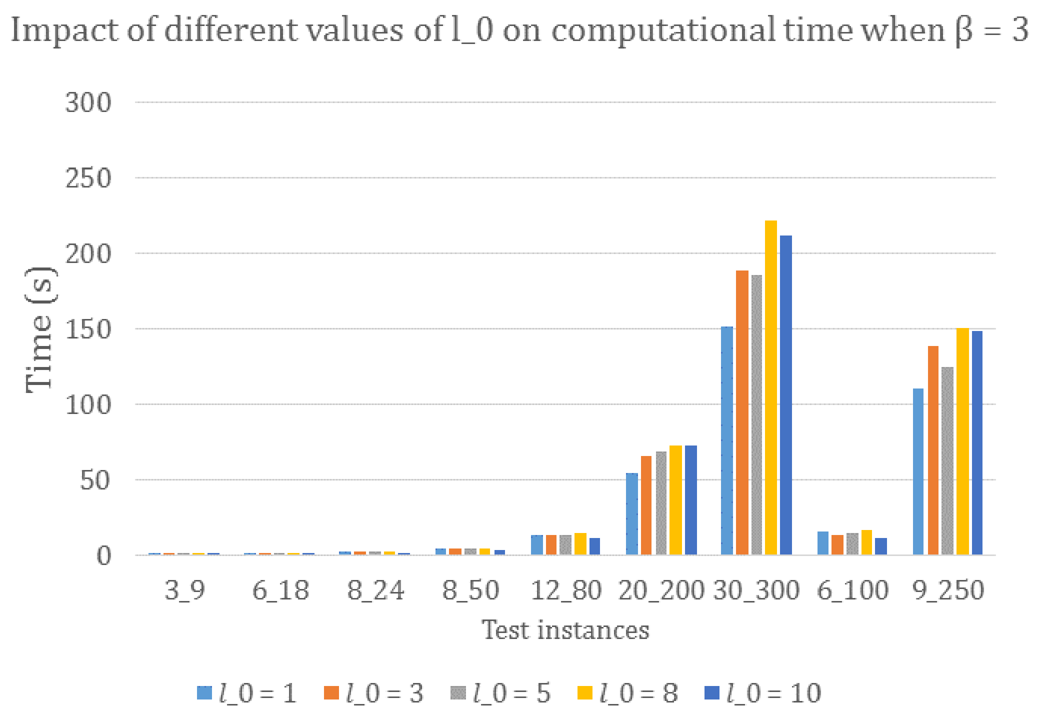
With the goal of studying the impact of parameter settings in LF-BAP, a statistical analysis is conducted in this subsection. In the Levy flight distribution, when the parameter $\beta$ increases the frequency of having long jumps decreases. Because the distribution is heavy tailed, the increase of $\beta$ 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 $\beta = 0.5, 1.5, 3, 5$.

We compare the runtime of different settings by allowing the algorithm to run until it converges to the same objective value. 9 random instances are chosen in different sizes among Set I, II, III and IV. A significant difference appears after conducting the

Friedman statistic test for all the combinations of $l_0$ and $\beta$. We notice a much larger runtime when $\beta = 0.5$ (Fig. 3.5). Excluding the combinations with $\beta = 0.5$, 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 are no significant differences between results from each group. Thus, the performance of LF-BAP is not noticeably sensitive to all the groups except for $\beta = 0.5$. The parameter setting of LF-BAP in the following experiments after the sensitivity analysis is shown in Table 3.4. All the settings of other algorithms are shown in the table as well.



(a)



(b)

FIGURE 3.5: The impact of different parameter settings on computational time with fixed $l_0$ and $\beta$, respectively.

Table 3.5: Summary of the comparison between LF-BAP and CPLEX-BAP

| 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% |

#### 3.3.3.3 Computational results

The experiments evaluate the performance of our algorithm in terms of accuracy, efficiency and capability. 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 (3.12) 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 H-BAP is also measured in the 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.

$$e = \frac{Avg.Obj_{LF} - Avg.Obj_{CPLEX}}{Avg.Obj_{CPLEX}} * 100\% \tag{3.12}$$

**Accuracy** In Table 3.5, if CPLEX-BAP cannot solve the problem due to the out-of-memory error while LF-BAP can solve it, LF-BAP is considered the faster algorithm and more accurate than CPLEX-BAP. According to Table 3.5, 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 errors are less than 1%. LF-BAP provides similar quality results (about 1% error) for solving most test cases in Set II, except for two instances with errors of 4.44% and 6.85%. In the results of large-scale problems (Tables 3.7 and 3.8), 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 than or equal to 5% is still above 60% for Set III and IV although it should be noted that in these sets there are only a few instances where errors can be determined thanks to CPLEX-BAP being able to solve them to optimality. In comparison with H-BAP, LF-BAP significantly outperforms H-BAP in all test cases in terms of objective values and relative errors by conducting a t-test (Tables 3.6, 3.7 and 3.8). All the negative percentages in these three tables indicate that LF-BAP obtains much better solutions than H-BAP.

Table 3.6: A comparison of average objective values and average computational time between H-BAP, 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 | | CPLEX-BAP | | | | LF-BAP | | Error between LF-BAP and CPLEX-BAP (%) | Error between LF-BAP and H-BAP (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 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 | 0.0872 | 1025 | 0.0022 | 469.8 | 0.0487 | 0.59% | -12.28% |
| | m=3, n=9 | big | 666.7 | 0.003 | 573.7 | 0.0427 | 1257 | 0.0001 | 587 | 0.0690 | 2.32% | -11.95% |
| | m=4, n=12 | small | 791.3 | 0.0036 | 717.4 | 0.101 | 2013.3 | 0.006 | 722.2 | 0.0510 | 0.67% | -8.73% |
| | m=4, n=12 | big | 811.7 | 0.003 | 712.5 | 0.0926 | 2158.1 | 0.006 | 718.8 | 0.0964 | 0.88% | -11.45% |
| | m=5, n=15 | small | 958.5 | 0.003 | 863.3 | 0.2814 | 2935.2 | 0.1324 | 869.5 | 0.2680 | 0.72% | -9.28% |
| | m=5, n=15 | big | 1066 | 0.0035 | 947.3 | 0.2419 | 2824.4 | 0.1109 | 954.1 | 0.3906 | 0.71% | -10.50% |
| | m=6, n=18 | small | 1081 | 0.0036 | 970.1 | 0.5698 | 3380.8 | 0.3738 | 973.3 | 0.4030 | 0.33% | -9.97% |
| | m=6, n=18 | big | 1215.2 | 0.003 | 1057.1 | 0.5939 | 3660.4 | 0.3749 | 1072.5 | 0.5715 | 1.46% | -11.74% |
| | m=7, n=21 | small | 1416.4 | 0.0031 | 1279.8 | 1.2425 | 4179.7 | 0.9485 | 1286.3 | 0.7554 | 0.51% | -9.19% |
| | m=7, n=21 | big | 1384.9 | 0.003 | 1195.6 | 1.2698 | 4355.9 | 0.0001 | 1201.8 | 1.2521 | 0.52% | -13.22% |
| | m=8, n=24 | small | 1577.2 | 0.003 | 1420 | 2.4369 | 5347.6 | 1.9069 | 1426.3 | 1.5097 | 0.44% | -9.57% |
| | m=8, n=24 | big | 1623.1 | 0.003 | 1394 | 2.1944 | 4561.8 | 1.7904 | 1411.4 | 1.8055 | 1.25% | -13.04% |
| II | m=6, n=30 | small | 1470.8 | 0.0038 | 1210.8 | 4.5912 | 7876.6 | 3.9872 | 1221.7 | 0.8000 | 0.90% | -16.94% |
| | m=6, n=30 | big | 1606.6 | 0.0106 | 1277.4 | 6.1932 | 7679 | 5.5132 | 1294 | 1.2092 | 1.30% | -19.45% |
| | m=7, n=40 | small | 3474 | 0.0046 | 2808.4 | 24.6882 | 14017.4 | 22.8302 | 2841.4 | 2.7945 | 1.18% | -18.21% |
| | m=7, n=40 | big | 2965 | 0.0062 | 2155.8 | 20.3352 | 12863.2 | 18.9092 | 2251.5 | 3.1968 | 4.44% | -24.06% |
| | m=8, n=50 | small | 2713.2 | 0.0064 | 2276.8 | 51.958 | 18654.8 | 47.954 | 2300.6 | 4.2861 | 1.05% | -15.21% |
| | m=8, n=50 | big | 4162.6 | 0.0046 | 3039 | 41.2578 | 15104 | 37.9658 | 3247.1 | 4.7728 | 6.85% | -21.99% |

Table 3.7: 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 | | CPLEX-BAP | | | | LF-BAP | | Error between LF-BAP and CPLEX-BAP (%) | Error between LF-BAP and H-BAP (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 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 | 5746.2 | 57.1016 | 19994.6 | 49.4976 | 5847.4 | 5.0637 | 1.76% | -15.56% |
| | m=9, n=60 | big | 6676.2 | 0.0052 | 5603.6 | 52.7044 | 20582.2 | 44.1524 | 5798.8 | 6.0131 | 3.48% | -13.14% |
| | m=10, n=70 | small | 8383.6 | 0.0086 | 7257.2 | 112.3554 | 24984.2 | 93.2554 | 7473.8 | 7.0482 | 2.98% | -10.85% |
| | m=10, n=70 | big | 8233.2 | 0.0058 | 6704.6 | 107.0084 | 27255.8 | 78.7364 | 6957.4 | 7.9936 | 3.77% | -15.50% |
| | m=12, n=80 | small | 8965.4 | 0.004 | 7724.8 | 673.6474 | 27838 | 665.5674 | 7911.6 | 11.0984 | 2.42% | -11.75% |
| | m=12, n=80 | big | 8986 | 0.0042 | 7589.4 | 2761.022 | 30329.2 | 2736.898 | 7901.9 | 11.7207 | 4.12% | -12.06% |
| | m=13, n=100 | small | 12262.6 | 0.0062 | 10325.6 | 3577.88 | 38603.8 | 3493.904 | 10599 | 14.1122 | 2.65% | -13.57% |
| | m=13, n=100 | big | 11673 | 0.0054 | 9977 | 2713.847 | 40098.4 | 2640.987 | 10316.7 | 16.2133 | 3.40% | -11.62% |
| | m=14, n=120 | small | 15581.2 | 0.0092 | N/S | N/S | N/S | N/S | 13835.1 | 25.1970 | N/S | -11.21% |
| | m=14, n=120 | big | 16724 | 0.0094 | N/S | N/S | N/S | N/S | 14421 | 28.0820 | N/S | -13.77% |
| | m=15, n=150 | small | 23094.8 | 0.0076 | N/S | N/S | N/S | N/S | 19904.8 | 35.6046 | N/S | -13.81% |
| | m=15, n=150 | big | 23457.8 | 0.0166 | N/S | N/S | N/S | N/S | 20215.8 | 38.9161 | N/S | -13.82% |
| | m=20, n=200 | small | 30812.4 | 0.018 | N/S | N/S | N/S | N/S | 27116.1 | 69.4433 | N/S | -12.00% |
| | m=20, n=200 | big | 30874.2 | 0.0102 | N/S | N/S | N/S | N/S | 26182.2 | 75.6344 | N/S | -15.20% |
| | m=30, n=300 | small | 45951.6 | 0.0166 | N/S | N/S | N/S | N/S | 39529.1 | 214.9714 | N/S | -13.98% |
| | m=30, n=300 | big | 47400.6 | 0.0152 | N/S | N/S | N/S | N/S | 40281.8 | 215.5681 | N/S | -15.02% |
| | m=40, n=400 | small | 61800.8 | 0.022 | N/S | N/S | N/S | N/S | 52936.2 | 468.3557 | N/S | -14.34% |
| | m=40, n=400 | big | 60219.6 | 0.0216 | N/S | N/S | N/S | N/S | 51751.4 | 438.4603 | N/S | -14.06% |
| | m=50, n=500 | small | 77196.8 | 0.0316 | N/S | N/S | N/S | N/S | 66371.4 | 652.9757 | N/S | -14.02% |
| | m=50, n=500 | big | 77988.8 | 0.0322 | N/S | N/S | N/S | N/S | 65460.2 | 955.2202 | N/S | -16.06% |

N/S: Not solvable by CPLEX-BAP.

Table 3.8: Computational results on large scale data Set IV.

In Set IV, each problem size (each row) contains 5 cases.

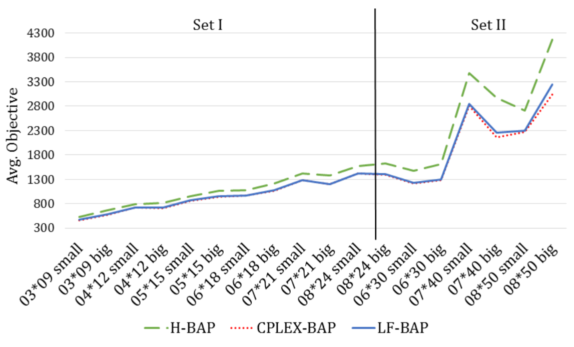
| Set | Problem size | Tidal effect | H-BAP | | CPLEX-BAP | | | | LF-BAP | | Error between LF-BAP and CPLEX-BAP (%) | Error between LF-BAP and H-BAP (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Avg. Obj. | Avg. Time (s) | Avg. Obj. | Avg. Time (s) | Avg.UB Obj. | Avg.UB Time (s) | Avg. Obj. | Avg. Time (s) | | |
| IV | m=5, n=80 | small | 20560.8 | 0.0208 | 17533.6 | 71.4994 | 54128.2 | 30.6754 | 18295.3 | 6.6913 | 4.34% | -11.02% |
| | m=5, n=80 | big | 22354.2 | 0.007 | 18048.2 | 58.1214 | 55926.2 | 32.5494 | 19465.6 | 9.2297 | 7.85% | -12.92% |
| | m=6, n=100 | small | 25114.4 | 0.0072 | 21001.4 | 169.952 | 66660.2 | 103.3 | 22131.7 | 11.9623 | 5.38% | -11.88% |
| | m=6, n=100 | big | 27525.6 | 0.0064 | 22512 | 519.1268 | 78162.6 | 443.9588 | 24464.4 | 15.5042 | 8.67% | -11.12% |
| | m=7, n=150 | small | 47946 | 0.0116 | N/S | N/S | N/S | N/S | 41710.2 | 31.9941 | N/S | -13.01% |
| | m=7, n=150 | big | 50884.6 | 0.012 | N/S | N/S | N/S | N/S | 44626.2 | 40.8227 | N/S | -12.30% |
| | m=8, n=200 | small | 77732.4 | 0.0234 | N/S | N/S | N/S | N/S | 68726.4 | 67.1615 | N/S | -11.59% |
| | m=8, n=200 | big | 78247.4 | 0.021 | N/S | N/S | N/S | N/S | 69696.7 | 77.2071 | N/S | -10.93% |
| | m=9, n=250 | small | 113770.8 | 0.0294 | N/S | N/S | N/S | N/S | 98179.3 | 123.0593 | N/S | -13.70% |
| | m=9, n=250 | big | 108725.4 | 0.0322 | N/S | N/S | N/S | N/S | 95080.4 | 141.5667 | N/S | -12.55% |
| | m=10, n=300 | small | 139798 | 0.0466 | N/S | N/S | N/S | N/S | 122812.5 | 199.9363 | N/S | -12.15% |
| | m=10, n=300 | big | 141726.8 | 0.0488 | N/S | N/S | N/S | N/S | 124669 | 229.4605 | N/S | -12.04% |

N/S: Not solvable by CPLEX-BAP.

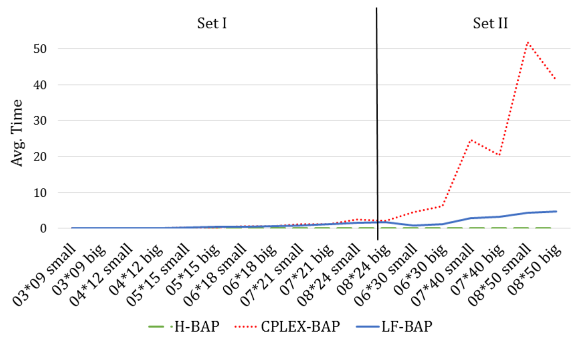
**Efficiency** All three algorithms start with a very small computational time (less than 1 second) according to Table 3.6. The meta-heuristic shows a much slower increase of computational time than the exact method when the problem size increases. (Fig. 3.6). To solve the largest problem size (m=8, n=50) in Set I and II, CPLEX-BAP takes 10 times the running time of LF-BAP. This becomes more obvious in the large-scale test cases in Table 3.7, 3.8 and Fig. 3.6. The running time of CPLEX-BAP rises to about 3,600 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 gradually gets noticeable (Fig. 3.6). The initial upper bounds and the time CPLEX took to find them is also reported in Tables 3.6, 3.7 and 3.8. 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 he 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 is significantly worse than the results by LF-BAP based on the t-test. 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 3.5 displays that in 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. 3.7 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. 3.7, the CV values are mostly less than 1%.

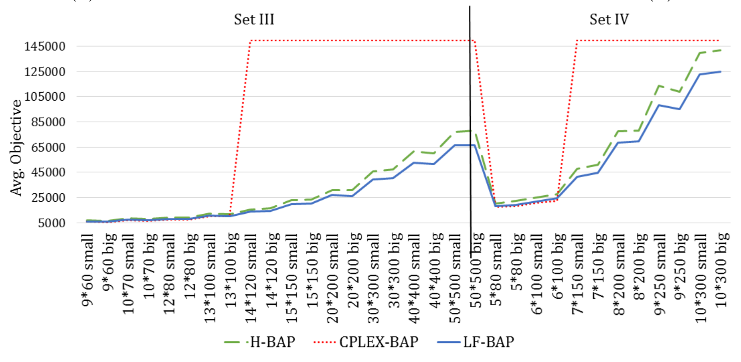
A majority of them are close to 0% indicating a very small difference between 50 runs. With a normal distribution assumed on the data of CV, [0.163%, 0.203%] is achieved as 95% confidence interval [Neyman, 1937] for the mean of CV. Therefore, a stable performance of our algorithm can be concluded due to the small interval of the mean of CV.



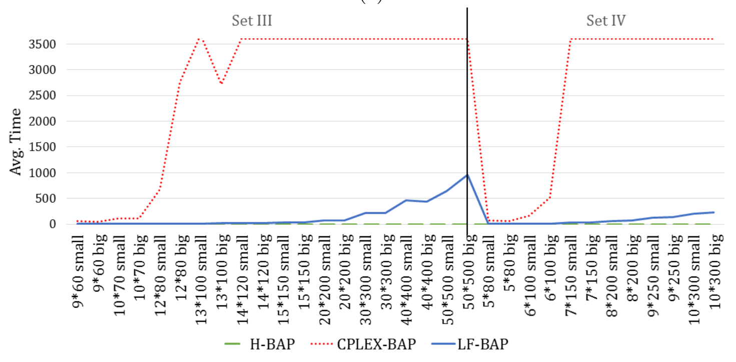
(a)



(b)



(c)



(d)

FIGURE 3.6: Comparison of three 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.

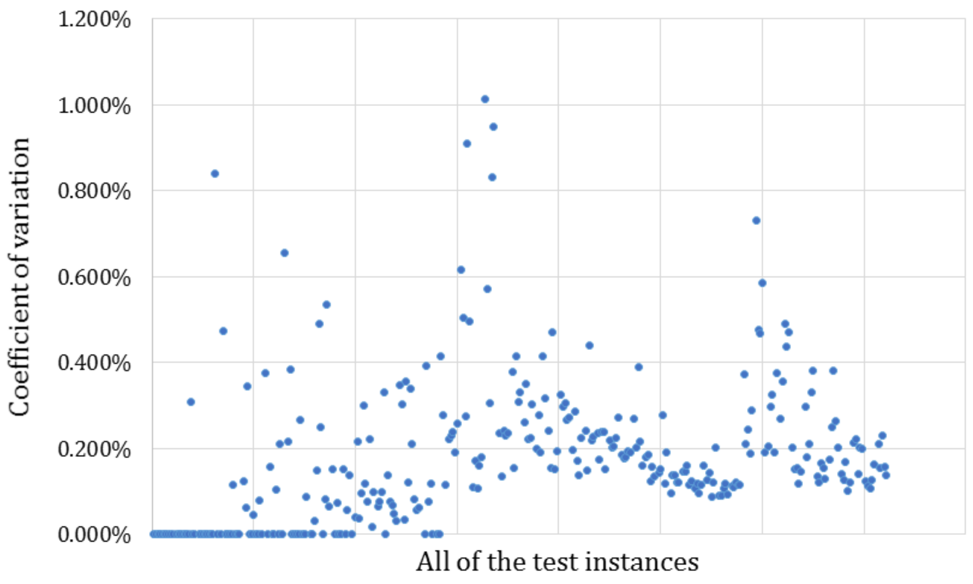


Figure 3.7: The coefficient of variation represents the robustness of LF-BAP

In summary, LF-BAP is better than 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%. In terms of computational cost, H-BAP has a very quick turnaround time for all the instances (less than 1 second). It can be seen from Fig. 3.6 (b and d) that comparing to H-BAP and CPLEX-BAP, the increase of runtime of LF-BAP is very slight 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. Compared 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 mentioning that CPLEX-BAP [Lalla-Ruiz et al., 2016] 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.

## 3.4 Conclusion

This chapter has made contributions as follows:

1. A new meta-heuristic based on Levy flight (LF-BAP) has been proposed to solve the Berth Allocation Problem taking tidal effect into consideration. As far as we know, this is the first time meta-heuristics have been used in solving BAPs with multi-tidal windows.
2. Large scale datasets were generated based on the instruction from Xu et al. [2012a]. The results of LF-BAP were compared with the state-of-the-art exact method using commercial solver CPLEX and a deterministic heuristic modified to fit multiple tides.
    - LF-BAP was faster than CPLEX model in 263 out of 310 cases. 60.97% of the cases LF-BAP either achieved a solution with similar quality (less than 0.5% error) of the global optimum or solved the problem while CPLEX could not.
    - LF-BAP was capable of finding feasible solutions for all test cases while CPLEX model was able to solve about 68% of them.
    - LF-BAP outperformed H-BAP in all test cases.
3. The sensitivity of parameters in LF-BAP is analysed.

Chapter 4

# Solving berth allocation problems with multi-tidal windows using Genetic algorithm

## 4.1 Introduction

In the last chapter, LF-BAP has proved competitive results in dealing with BAPs while taking multiple tides into account. However, in terms of the quality of solutions, there are some cases in which the remarkable CPLEX model outperforms LF-BAP. In this chapter, we will propose a population-based meta-heuristic in order to further investigate the problem. By introducing another algorithm, we will also try to reduce the gap between the exact technique and the approximate method.

In the following section, the proposed GA is introduced in detail and experiments comparing the GA with the state-of-the-art algorithms are conducted. Moreover, an investigation of how meta-heuristics perform on this problem is carried out by conducting experiments among LF-BAP, PSO and GA in Section 4.3, followed by a conclusion of this chapter in Section 4.4.

## 4.2 Genetic algorithm for BAPs

With the goal of investigating the performance of meta-heuristics on BAPs, we also propose a population-based meta-heuristic in this chapter. Unlike single-point search methods which improve one solution by exploring neighbourhood while maintaining a diversity, population-based meta-heuristics explore the search space by the cooperation between population elements and exploit the collected information to reach potential solutions by a competition between the population elements.

In this section, a GA is proposed to study the best way of scheduling vessels in a port. GAs have been widely studied in Michalewicz and Schoenauer [1996]. The GA is a nature inspired algorithm that belongs to the class of evolutionary algorithms. It was invented with the idea of using the power of evolution to solve optimisation problems. GAs work by evolving a set of individuals towards better solutions. A set of individuals are known as a population. Each individual has a chromosome carrying its own information. The concept of GAs is improving the chromosomes of the population by doing permutations along a number of iterations (called generations). GAs have been proved to have a robust performance under a reasonable level of noise. They do not break easily even if the inputs are changed slightly. Reflecting real-world scenarios, even if the actual arrival time of a vessel changes slightly from the estimated arrival time, the overall cost of the schedule is unlikely to significantly increase.

### 4.2.1 Chromosome representation

According to existing GAs in solving BAPs, there are two common ways to represent the chromosome. The first one is representing the chromosome as a berthing schedule directly [Theofanis et al., 2007, Golias et al., 2009a, Golias and Haralambides, 2011]. Once a small change is made to the schedule, it can cause extra delay for some of the vessels. Assume a vessel is moved to an earlier time, one of the following vessels now has to wait until a high tide and all the vessels after will have to wait extra time. This also makes the berth idle for a low tide period, wasting valuable time that could be used to accommodate other vessels.

To overcome the inefficient way of doing permutations, the priority of allocating vessels is suggested in Hu [2015], Lalla-Ruiz et al. [2014]. Lalla-Ruiz et al. [2014] represented the chromosomes as vectors where the berthing order is defined by the first part and the quay crane assignment is held by the second part. The biased random key generates numbers in [0, 1). By doing vector computations on real values, the berthing order is decided. Hu [2015] used integers to represent the priority of processing vessels. The algorithm firstly considers the vessel with highest priority based on the chromosome. The vessel is sent to a berth with the goal of maintaining the minimum increment of cost.

We agree that solving BAPs with multiple tidal windows can benefit from representing the chromosome as an order of processing vessels. Unlike the way mentioned above, the newly proposed chromosome is a combination of vessel orders and berth orders. The priority of processing vessels is determined by multiple sequences. Once the order of processing vessels is decided, we follow the pre-set rules in the deterministic heuristic from Xu et al. [2012a] in order to have the berthing schedule.

The greedy heuristic introduced in Xu et al. [2012a] allocates vessels in the order of Weighted Shortest Processing Time (WSPT). For each vessel, the heuristic chooses a berth with the minimum increment of the objective value while taking into account the availability of berths at current tide. 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 Xu et al. [2012a] to improve the only solution. In our case, following the rules in Xu et al. [2012a], the final schedule is generated based on the order of vessels in the chromosome instead of WSPT.

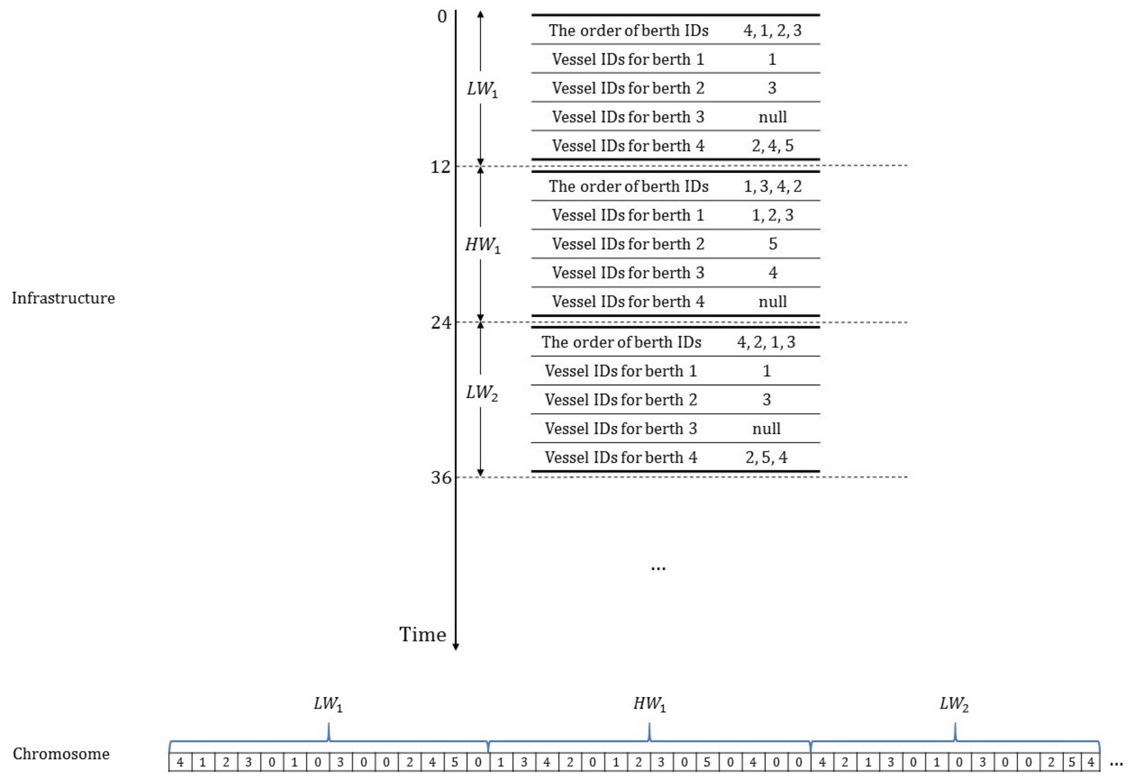
An example of the chromosome is shown in Table 4.1. The first line is the order of berths where the priority of processing each group of vessels is decided. Regarding the groups of vessels, vessels with the same berth indicator (see Table 3.1) grouped together. In the example (Table 4.1), vessels with berth ID = 4 will be assigned first, followed by the group with berth ID = 1. Reflecting to vessels, vessel IDs = 2, 4, 5 are processed first followed by vessel IDs = 1. Neither the vessel IDs nor the berth IDs can be swapped between lines because it is likely that a berth is not available for a randomly swapped vessel.

It is unlikely that all vessels fit in one low tide and one high tide, especially when the port is busy. If we use the same orders repeatedly, a swap in the chromosome affects the processing order at more than one low tide or high tide. A major change can be made

Table 4.1: An example of the chromosome in GA

| | |
|---|---|
| The order of berth IDs | 4, 1, 2, 3 |
| Vessel IDs for berth 1 | 1 |
| Vessel IDs for berth 2 | 3 |
| Vessel IDs for berth 3 | null |
| Vessel IDs for berth 4 | 2, 4, 5 |

An example of the chromosome for one tide



| | |
|---|---|
| The order of berth IDs | 4, 1, 2, 3 |
| Vessel IDs for berth 1 | 1 |
| Vessel IDs for berth 2 | 3 |
| Vessel IDs for berth 3 | null |
| Vessel IDs for berth 4 | 2, 4, 5 |

| | |
|---|---|
| The order of berth IDs | 1, 3, 4, 2 |
| Vessel IDs for berth 1 | 1, 2, 3 |
| Vessel IDs for berth 2 | 5 |
| Vessel IDs for berth 3 | 4 |
| Vessel IDs for berth 4 | null |

| | |
|---|---|
| The order of berth IDs | 4, 2, 1, 3 |
| Vessel IDs for berth 1 | 1 |
| Vessel IDs for berth 2 | 3 |
| Vessel IDs for berth 3 | null |
| Vessel IDs for berth 4 | 2, 5, 4 |

An example of the chromosome for multiple tides

to the schedule by a small move. To avoid causing too frequent long jumps, we replicate an independent order like Table 4.1 for each tidal period, instead of using the same one repeatedly for all the tides. In other words, our chromosome is a combination of the orders for each tide. For example in Table 4.1, at low tides $LW_1$ and $LW_2$ , the orders of berths and vessels are independent but the group to which each vessel belongs stays the same. A large number of combinations is necessary to ensure that all the vessels are scheduled along the time horizon while the tide changes.

The main reasons for proposing this new chromosome representation are: 1) having both berth IDs and vessel IDs in the chromosome and different orders in each tide maintains the diversity of the search space; 2) when some berths are not available to some vessels while the tide is changing, Xu et al. [2012a] always guarantees feasible solutions; 3) by attempting feasible solutions in the search space, Xu et al. [2012a] is able to find the best solution when inserting a new vessel to the current schedule.

### 4.2.2 Description of the GA procedure

With the chromosome representation we have proposed in the previous section, the procedure of our GA is summarised in Algorithm 5. The population *Pop* in the algorithm contains $N$ individuals. In each generation *Pop* is evolved by selecting the best $N$ from three groups of individuals: current *Pop*, the mutation of *Pop*, and the offspring of *Elites* and *Pop*.

GAs are well-known as a time-consuming approach. The following stopping criteria are applied to our GA to accelerate the process: 1) a predefined maximum number of generations is reached; 2) a predefined maximum number of consecutive generations without improvement; 3) the known global optimum is reached (a global optimum can be found by the exact technique from the commercial solver, if it is able to solve the problem).

**Algorithm 5** The pseudocode of GA

```
Let $Pop$ be the population and $N$ be the size of the population.
Let $M$ be the size of the elitism.
$Pop$ := Initialisation().
While (stopping_criteria_not_met) do:
    Denote $Elites$ the best $M$ of $Pop$.
    $Pop_1$ := Mutation ($Pop$).
    Denote the offspring of crossover $Pop_2$.
    For i := 1 to $N$ do:
        $Parent_1$:= Randomly pick from $Elites$.
        $Pop_2[i]$:= Crossover ($Parent_1$,TournamentSelection($Pop$)).
    End
    Decode the chromosomes (Algorithm 9) and apply an intensification to each solution (Algorithm 10).
    Evaluate and select the best $N$ from $Pop$, $Pop_1$ and $Pop_2$, and denote $Pop'$.
    $Pop$ := $Pop'$.
End
```

#### 4.2.2.1 Initialisation

In Algorithm 6, we initialise one individual in the population following rules in Step 1. When initialising the rest of the individuals, the group to which each vessel belongs stays the same. We randomise the order of processing vessels within a group instead of using the ascending order of weighted processing time (Step 2 in Algorithm 6).

**Algorithm 6** Initialisation steps

```
Step 1 Initialise one individual in the population:
        The sequence of berths is in ascending order of water-depths.
        For each line of vessels, sort by the weighted processing time $t\_proc_j/w_j$.
        The combination is replicated for each tidal period.
Step 2 Initialise the rest of the population:
        Randomise the order of berth.
        Randomise the order of each group of vessels.
```

#### 4.2.2.2 Elitism strategy

The elites are formed from a number of solutions with the best fitness values. They are updated in every generation. In Crossover and Tournament selections, we combine the elites with other individuals from the population to produce offspring.

#### 4.2.2.3 Mutation

We apply two different rules to mutations of the order of berths and vessels respectively. For each berth order, if the mutation rate is satisfied we randomly swap two berths (Fig. 4.1). In the mutation of vessel orders, we regenerate the order randomly (Fig. 4.2).

#### 4.2.2.4 Crossover and Tournament selection

In crossover, two parents are needed. One is randomly selected from the elites as a parent, and another one is chosen by applying a tournament selection (Algorithm 8). A number of tournaments are randomly picked from the current population. And then we choose the best performing one from tournaments as another parent. Once the parents are decided, we combine different lines from the parents in crossover (Algorithm 7). In order to form a new chromosome, we take a line from one parent with a certain possibility $p$ otherwise we take it from another parent.

**Algorithm 7** The pseudocode of *Crossover*

```
Let $L$ be the total number of lines of the chromosome.
Let $Child$ be the output of the crossover.
Let $Parent_1$, $Parent_2$ denote the two parents of the offspring $Child$.
Let $CrossoverRate$ denote the ratio of inheriting from $Parent_1$ and $Parent_2$
For i := 1 to $L$ do:
  Denote$Parent_1[i]$ and $Parent_2[i]$ the current line of each parent.
  Randomise a possibility $p$.
  If ( $p \leq CrossoverRate$ ) do:
    $Child_i := Parent_1[i];$
  Else
    $Child_i := Parent_2[i];$
  End
End
```

**Algorithm 8** The pseudocode of *TournamentSelection*

```
For i := 1 to $TournamentSize$ do:
  Denote $Tournament_i$ a randomly selected individual from the population.
  Let $Obj_i$ be the fitness value of $Tournament_i$.
End
Choose the individual $Tournament_B$ with the best fitness value $Obj_B$.
```

FIGURE 4.1: Mutation for berths

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| The order of berth IDs | **4** | 1 | **2** | 3 | $\Longrightarrow$ | The order of berth IDs | **2** | 1 | **4** | 3 |

FIGURE 4.2: Mutation for vessels

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vessel IDs for berth i | 2 | 4 | 8 | 7 | 1 | $\Longrightarrow$ | Vessel IDs for berth i | 1 | 7 | 4 | 2 | 8 |

#### 4.2.2.5 Decoding and intensification

Before evaluating the performance of the population, firstly we decode the chromosome. Algorithm 9 describes the rules of transforming the chromosome to a schedule. The scheduling process assigns vessels to one tidal period and then to another by following Xu et al. [2012a] until all the vessels have been assigned. The outcome schedule includes the time and berth to serve each vessel.

With the achieved schedule, we apply an intensification based on the current schedule in order to find the local optimum. This step aims to improve solutions by proposing a new neighbourhood structure. Each individual explores its neighbouring solutions by firstly applying internal operations and then external operations. Three neighbouring solutions are found by applying internal operations and the one with the best improvement of fitness value will be kept. External operations explore neighbouring solutions by moving vessels between berths. The detail is explained in Algorithm 10.

**Algorithm 9** The pseudocode of the decoding procedure

```
Let $OrgSol$ be the solution to be decoded.
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 3.2, $L_j$ represents the set of available berths for vessel j is $\{L_j, L_j+1, ..., m\}$ at low tide.
Let $H$ be a set of available berths for n vessels at high tide $H := \{H_1, H_2, ..., H_n\}$. //$H_j$ represents the set of available berths for vessel j is $\{H_j, H_j+1, ..., m\}$ at high tide.
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
  Denote $B_{Seq}$ the berth sequence of $OrgSol$ at this tide.
  Let $V$ be the vessel sequence at this tide. $V := \{OrgSol_{B_{Seq_1}}, OrgSol_{B_{Seq_2}},..., OrgSol_{B_{Seq_m}}\}$. $OrgSol_{B_{Seq_1}}$ indicates a sequence of vessels the 1st berth in $B_{Seq}$ corresponds to.
  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.
      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 10** The pseudocode of *NeighbourhoodStructure*

```
Let $S$ be the current schedule.
Let $B$ be a set of $m$ berths $B$ := {1, 2, ..., m}.
//internal operations
For $i$:= 1 to m do:
  Let $A := S_{B_i}$ denote the list of vessels sent to $B_i$.
  Find the vessel $A_j$ with the largest cost $w_{Aj} * t\_wait_{A_j}$ and denote its arrival
  time $T_{A_j}$.
  //N1
  If time $T_{A_j}$ is idle do: Let $p_1$ denote the position.
  Else do:
    Find the position after the vessel occupying time $T_{A_j}$ and denote $p_1$.
  End
  Insert $A_j$ to $p_1$ and denote the solution as $N_1$.
  //N2
  From $p_1$ in $A$ find the first vessel with a waiting time ¿ 0 and let the position
  be $p_2$.
  Swap $A_j$ and $A_{p_2}$ and then let $N_2$ denote the new solution.
  //N3
  Swap $A_j$ and $A_{j-1}$ and let $N_3$ denote the new solution.
  Evaluate $N_1$, $N_2$ and $N_3$ and update $S_{B_i}$with the one with the best improve-
  ment.
End
//external operations. N4 swap 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 $S_{B_j}$.
    Switch $V_{B_i,q}$ 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
//N5 move vessels from one berth to another
For $i$:= m to 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 1 do:
    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
```

TABLE 4.2: GA with different operations

| Name | Operators |
|---|---|
| OP1 | Mutation |
| OP2 | Crossover with tournament selection |
| OP3 | Mutation and crossover with tournament selection |
| OP4 | Mutation, crossover with tournament selection and neighbourhood structure |

TABLE 4.3: List of instances tested for GA with different operators and their combinations

| Instance Name | Berth No. | Vessel No. | Tidal Effect |
|---|---|---|---|
| Instance A | 6 | 18 | big |
| Instance B | 8 | 50 | big |
| Instance C | 10 | 70 | small |
| Instance D | 5 | 80 | small |

### 4.2.3 Computational experiments

#### 4.2.3.1 An analysis of GA with multiple operators

In this section, we analyse the effects of GA with different operators and their combinations. Mutation, crossover with tournament selection and neighbourhood search are investigated (OP1 OP4 in Table 4.2). Four representative instances are tested (Table 4.3). The experimental results in terms of generation numbers and objective values are displayed in Figs. 4.3 and 4.4. It is noticeable that: 1) OP4 obtains significantly better objective value than the other three in all instances we have tested; 2) OP4 also converges fairly quickly; 3) OP2 takes the least number of generations to converge but it does not reach good-quality solutions; 4) OP1 and OP3 perform similarly and sometimes OP1 obtains better solutions. It means that the crossover may not have direct impact on finding good-quality solutions but it helps the algorithm to quickly converge.

#### 4.2.3.2 Comparison with existing work

We compare the performance of the proposed GA with the following algorithms: a state-of-the-art exact method called Generalised Set-Partitioning BAP [Lalla-Ruiz et al., 2016] (CPLEX-BAP) and a greedy heuristic [Xu et al., 2012a] (H-BAP).

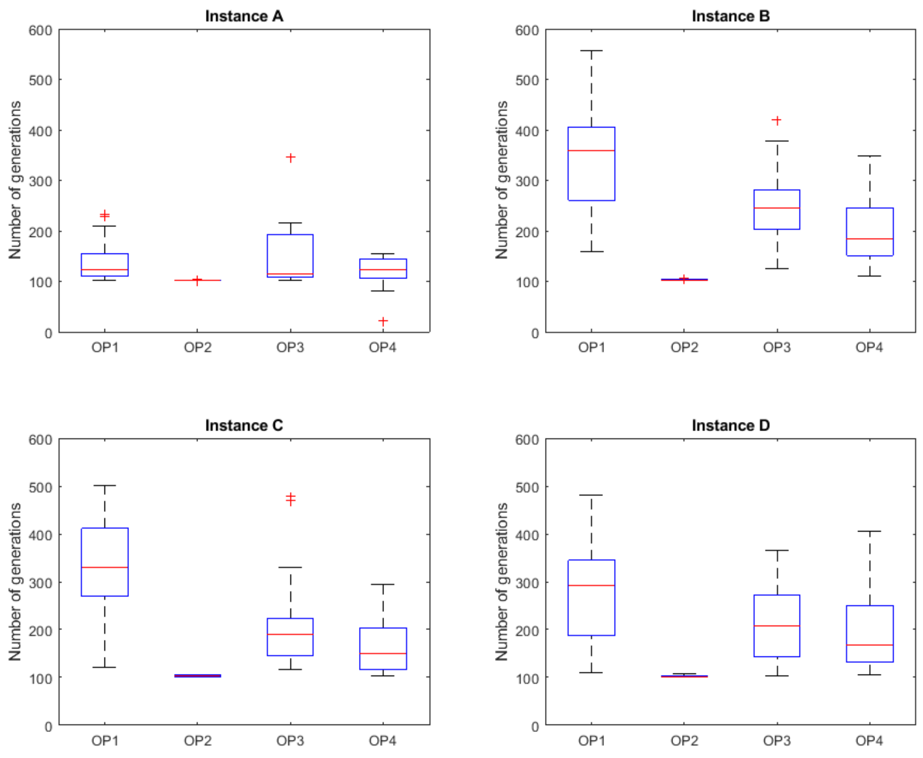


FIGURE 4.3: The distribution of the total number of generations taken by GA with different operators.
20 runs are conducted for each instance with following parameters: mutation rate = 0.3, crossover rate = 0.5, tournament size = 5, population size = 20, elitism size = 10. The algorithm stops if one-hour time limit is reached or the solution is not improved in continuous 100 generations.

H-BAP is also modified to accomodate multiple tides since it only takes into account two tidal windows in Xu et al. [2012a]. The original algorithm from Xu et al. [2012a] is deterministic. The available berth with the least cost is allocated to each vessel. The order of scheduling vessels is based on the weighted processing time. We limit the second tidal period to the same length as the first one and then repeat the process until all the vessels are scheduled. By comparing with H-BAP, we are able to identify how much the stochastic elements enhance the performance.

The same relative error $e$ (3.12) is used in order to compare an algorithm with the global optima (found by the exact technique in CPLEX_BAP). $e$ quantifies the difference between two algorithms regarding objective values.

An overview in Table 4.4 shows that 73.8% runs of our algorithm are able to find a

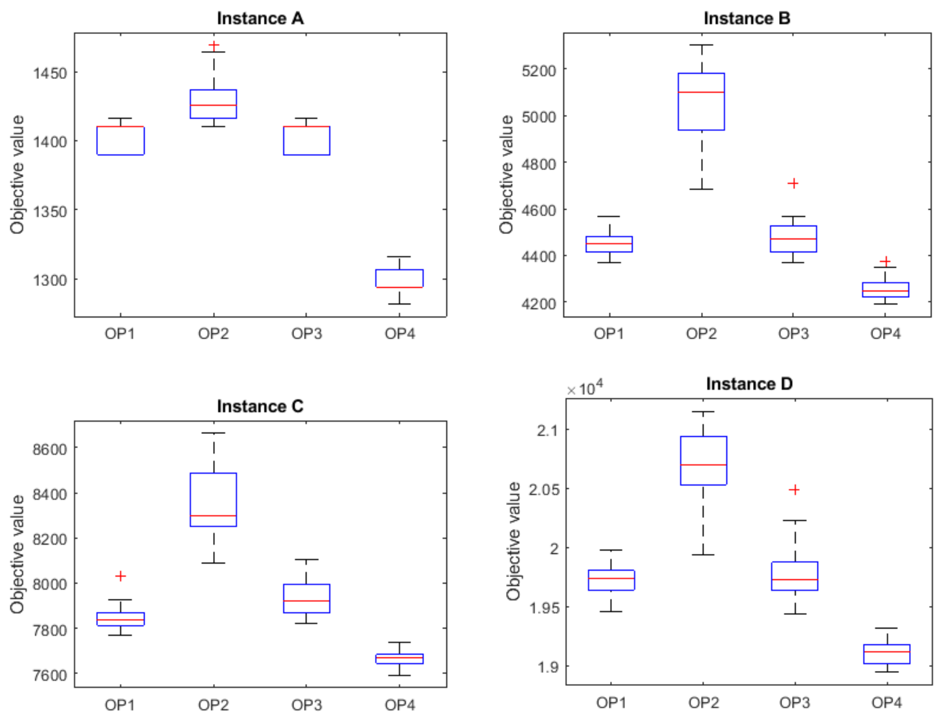


FIGURE 4.4: The distribution of the objective value achieved by GA with different operators.

solution either with $e \leq 0.5\%$ or that CPLEX-BAP cannot solve among all 310 test instances. For small-scale data sets, 96.5% in Set I and 42.3% in Set II are solved by GA with $e \leq 0.5\%$ (Table 4.5). In terms of runtime, GA achieves the optima more quickly than CPLEX-BAP in all test cases of Set I and II. GA keeps the runtime less than 1 second in Set I and at most 8.9 seconds in Set II. CPLEX-BAP has a short turnaround time (less than 2 seconds) in Set I and it increases in Set II up to 50 seconds approximately.

The relative error of GA slightly increases when solving large-scale problems (Table 3.7) but the relative error of the majority is still within 5% for cases CPLEX-BAP is able to solve. As the level of the problem complexity gets higher, all the algorithms take longer to converge. The running time of CPLEX-BAP increases exponentially due to the limitation of exact techniques while GA takes from 10 seconds and up to about 1,700 seconds for all the test cases. For the most complex case CPLEX-BAP could solve, it takes CPLEX-BAP and GA about 3,600 seconds and 68 seconds respectively. Because our algorithm is terminated when it reaches the pre-set maximum number of non-improving consecutive generations, GA is showing a quicker convergence in complicated problems. This is demonstrated by a decrease in running time when the problem scale gets to 30 berths and 300 vessels.

Table 4.4: Summary of the comparison between GA and CPLEX-BAP

| Data set | No. of test cases | Solvable cases | | Faster algorithm | | Percentage of cases with an error (%) $e$ between GA and CPLEX-BAP | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | CPLEX-BAP | GA | CPLEX-BAP | GA | $e \leq 0.5\%$ | $0.5\% < e \leq 1\%$ | $1\% < e \leq 2\%$ | $2\% < e \leq 5\%$ | $e > 5\%$ |
| I | 120 | 120 | 120 | 0 | 120 | 96.5% | 2.6% | 0.9% | 0 | 0 |
| II | 30 | 30 | 30 | 0 | 30 | 42.3% | 17.7% | 18.5% | 20.8% | 0.7% |
| III | 100 | 40 | 100 | 0 | 100 | 60.1% | 1.6% | 27.9% | 10.4% | 0 |
| IV | 60 | 20 | 60 | 0 | 60 | 66.7% | 0 | 4.4% | 16.8% | 12.1% |
| Total | 310 | 210 | 310 | 0 | 310 | 73.8% | 3.2% | 12% | 8.6% | 2.4% |

Time limitation of CPLEX-BAP and GA is 1 hour.
We run GA 20 times on each instance.
Maximum number of function evaluations = 10000; maximum number of non-improving consecutive generations = 1000.
The chromosome size = 20; crossover rate = 0.5; mutation rate = 0.3; elitism size = 10; tournament size = 5.

Table 4.5: A comparison of average objective values and average computational time between H-BAP, CPLEX-BAP and GA on Set I and II

| Set | Problem size | Tidal effect | H-BAP | | CPLEX-BAP | | GA | | Error between GA and CPLEX-BAP (%) |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg. Obj. | Avg. Time (s) | Avg. Obj. | Avg. Time (s) | Avg. Obj. | Avg. Time (s) | |
| I | m=3, n=9 | small | 535.5 | 0.0043 | 467 | 0.0872 | 467.1 | 0.0095 | 0.01% |
| | m=3, n=9 | big | 666.7 | 0.003 | 573.7 | 0.0427 | 573.9 | 0.0146 | 0.05% |
| | m=4, n=12 | small | 791.3 | 0.0036 | 717.4 | 0.101 | 719.3 | 0.0472 | 0.21% |
| | m=4, n=12 | big | 811.7 | 0.003 | 712.5 | 0.0926 | 714 | 0.0583 | 0.23% |
| | m=5, n=15 | small | 958.5 | 0.003 | 863.3 | 0.2814 | 864.1 | 0.0186 | 0.10% |
| | m=5, n=15 | big | 1066 | 0.0035 | 947.3 | 0.2419 | 948.4 | 0.0384 | 0.12% |
| | m=6, n=18 | small | 1081 | 0.0036 | 970.1 | 0.5698 | 972.3 | 0.2209 | 0.22% |
| | m=6, n=18 | big | 1215.2 | 0.003 | 1057.1 | 0.5939 | 1061.1 | 0.2516 | 0.35% |
| | m=7, n=21 | small | 1416.4 | 0.0031 | 1279.8 | 1.2425 | 1284.5 | 0.2446 | 0.37% |
| | m=7, n=21 | big | 1384.9 | 0.003 | 1195.6 | 1.2698 | 1200 | 0.5325 | 0.37% |
| | m=8, n=24 | small | 1577.2 | 0.003 | 1420 | 2.4369 | 1425.5 | 0.5343 | 0.39% |
| | m=8, n=24 | big | 1623.1 | 0.003 | 1394 | 2.1944 | 1402 | 1.1626 | 0.58% |
| II | m=6, n=30 | small | 1470.8 | 0.0038 | 1210.8 | 4.5912 | 1218.6 | 1.1645 | 0.48% |
| | m=6, n=30 | big | 1606.6 | 0.0106 | 1277.4 | 6.1932 | 1286.2 | 1.0782 | 0.65% |
| | m=7, n=40 | small | 3474 | 0.0046 | 2808.4 | 24.6882 | 2823.7 | 3.0369 | 0.60% |
| | m=7, n=40 | big | 2965 | 0.0062 | 2155.8 | 20.3352 | 2199.2 | 3.9543 | 1.91% |
| | m=8, n=50 | small | 2713.2 | 0.0064 | 2276.8 | 51.958 | 2294.2 | 4.9517 | 0.71% |
| | m=8, n=50 | big | 4162.6 | 0.0046 | 3039 | 41.2578 | 3124.4 | 8.9558 | 2.76% |

Table 4.6: A comparison of average objective values and average computational time between H-BAP, CPLEX-BAP and GA on Set III and IV

| Set | Problem size | Tidal effect | H-BAP | | CPLEX-BAP | | GA | | Error between GA and CPLEX-BAP (%) |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg. Obj. | Avg. Time (s) | Avg. Obj. | Avg. Time (s) | Avg. Obj. | Avg. Time (s) | |
| III | m=9, n=60 | small | 6924.6 | 0.014 | 5746.2 | 57.1 | 5840.5 | 10.3 | 1.65% |
| | m=9, n=60 | big | 6676.2 | 0.0052 | 5603.6 | 52.7 | 5733.5 | 12.8 | 2.32% |
| | m=10, n=70 | small | 8383.6 | 0.0086 | 7257.2 | 112.4 | 7403.2 | 16.7 | 2.01% |
| | m=10, n=70 | big | 8233.2 | 0.0058 | 6704.6 | 107 | 6952.5 | 19.6 | 3.53% |
| | m=12, n=80 | small | 8965.4 | 0.004 | 7724.8 | 673.6 | 7881.2 | 33.2 | 1.99% |
| | m=12, n=80 | big | 8986 | 0.0042 | 7589.4 | 2761 | 7866.5 | 31.4 | 3.57% |
| | m=13, n=100 | small | 12262.6 | 0.0062 | 10325.6 | 3677.9 | 10588.6 | 68.6 | 2.54% |
| | m=13, n=100 | big | 11673 | 0.0054 | 9977 | 2713.8 | 10283.1 | 59.6 | 3.05% |
| | m=14, n=120 | small | 15581.2 | 0.0092 | N/S | N/S | 13794.7 | 109 | N/S |
| | m=14, n=120 | big | 16724 | 0.0094 | N/S | N/S | 14355.9 | 107.1 | N/S |
| | m=15, n=150 | small | 23094.8 | 0.0076 | N/S | N/S | 19908.8 | 223.3 | N/S |
| | m=15, n=150 | big | 23457.8 | 0.0166 | N/S | N/S | 20156.9 | 239.1 | N/S |
| | m=20, n=200 | small | 30812.4 | 0.018 | N/S | N/S | 27182 | 1405.4 | N/S |
| | m=20, n=200 | big | 30874.2 | 0.0102 | N/S | N/S | 26213 | 1400.2 | N/S |
| | m=30, n=300 | small | 45951.6 | 0.0166 | N/S | N/S | 40009.5 | 759.1 | N/S |
| | m=30, n=300 | big | 47400.6 | 0.0152 | N/S | N/S | 40509.2 | 758 | N/S |
| | m=40, n=400 | small | 61800.8 | 0.022 | N/S | N/S | 53665.2 | 790.5 | N/S |
| | m=40, n=400 | big | 60219.6 | 0.0216 | N/S | N/S | 52320.9 | 803.2 | N/S |
| | m=50, n=500 | small | 77196.8 | 0.0316 | N/S | N/S | 67736.8 | 1026 | N/S |
| | m=50, n=500 | big | 77988.8 | 0.0322 | N/S | N/S | 66504.7 | 947 | N/S |

N/S: Not solved

Table 4.7: A comparison of average objective values and average computational time between H-BAP, CPLEX-BAP and GA on Set IV

| Set | Problem size | Tidal effect | H-BAP | | CPLEX-BAP | | GA | | Error between GA and CPLEX-BAP (%) |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg. Obj. | Avg. Time (s) | Avg. Obj. | Avg. Time (s) | Avg. Obj. | Avg. Time (s) | |
| IV | m=5, n=80 | small | 20560.8 | 0.0208 | 17533.6 | 71.5 | 18082.9 | 16.6 | 3.08% |
| | m=5, n=80 | big | 22354.2 | 0.007 | 18048.2 | 58.1 | 19012.1 | 19.6 | 5.27% |
| | m=6, n=100 | small | 25114.4 | 0.0072 | 21001.4 | 169 | 21869.2 | 31.2 | 4.11% |
| | m=6, n=100 | big | 27525.6 | 0.0064 | 22512 | 519.1 | 23929.2 | 33.2 | 6.34% |
| | m=7, n=150 | small | 47946 | 0.0116 | N/S | N/S | 41596.7 | 128.2 | N/S |
| | m=7, n=150 | big | 50884.6 | 0.012 | N/S | N/S | 44122.4 | 121 | N/S |
| | m=8, n=200 | small | 77732.4 | 0.0234 | N/S | N/S | 68210.9 | 343.8 | N/S |
| | m=8, n=200 | big | 78247.4 | 0.021 | N/S | N/S | 68420.9 | 321.7 | N/S |
| | m=9, n=250 | small | 113770.8 | 0.0294 | N/S | N/S | 97847.2 | 725.4 | N/S |
| | m=9, n=250 | big | 108725.4 | 0.0322 | N/S | N/S | 94008.2 | 1563.3 | N/S |
| | m=10, n=300 | small | 139798 | 0.0466 | N/S | N/S | 122557.6 | 1682.9 | N/S |
| | m=10, n=300 | big | 141726.8 | 0.0488 | N/S | N/S | 123241.5 | 1595 | N/S |

N/S: Not solved

In summary, the proposed GA has shown competitive performance in solving BAPs with tidal constraints. When compared with the remarkable exact method, GA produces good solutions with reasonable gaps. Even in many small-scale cases, GA is able to achieve the global optima with a shorter computational time. For large-scale data sets, GA obtains better results than all the other algorithms we compare, while CPLEX-BAP is not able to obtain a solution due to the memory error for most of the instances. It leads GA to be a good option to solve this problem especially in practical busy terminals, where a quick responding solution is desired.

In this section, a new GA is proposed to solve BAPs while considering multiple tides. A new neighbourhood structure is also proposed to enhance the performance. According to the experiments, this approach also produces competitive outcomes compared to the state-of-the-art techniques. We believe the results have been promising to significantly improve the efficiency of port operations in practical situations. In the next section, we will further compare the performance of two proposed algorithms and another existing meta-heuristic.

## 4.3 Study of meta-heuristics on BAPs

GA and LF-BAP are compared in this section to study their performance on BAPs with multi-tidal windows. A PSO from Ting et al. [2014] is also replicated for the assessment. As far as we know, there is no existing meta-heuristic dealing with discrete dynamic BAPs with multi-tidal windows. Approximate methods that we have reviewed consider different problem settings and none of them is able to be modified to solve BAPs with tidal constraints. The PSO is replicated and the evaluation function is modified so that it fits multiple-tidal windows. In the evaluation function of PSO we also 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.

The algorithms are run until they converge. We run each of them 20 times. If they are not improved in 1000 number of function evaluations, they are identified as converged and terminated. Tables 4.8, 4.9 and 4.10 record the average number of function evaluations taken to convergence by each algorithm and its solution quality.

GA and LF-BAP both outperform PSO in all test cases. Comparing two proposed meta-heuristics, it is noticeable that GA performs better for all test cases in Set I, II and IV. The difference is big for some cases eg. m=8, n=200 with big tidal effect. For instances of Set III, LF-BAP gets better than GA when the number of berths m $\geq$ 15. In terms of the speed of convergence, GA converges significantly faster than LF-BAP in all instances which is proved by a t-test.

In summary, for small scale cases, GA effectively achieves better-quality solutions compared to LF-BAP. It happens to most of the large-scale cases as well, such as Set IV. For some cases in Set III (m $\geq$ 30), it can be seen that GA converges in a few hundreds of function evaluations but sacrificing a little in the quality of solutions, in which LF-BAP takes more than 3,000 function evaluations.

Table 4.8: A comparison between LF-BAP, GA and PSO on Set I and II

| Set | Problem size | Tidal effect | LF-BAP | | GA | | PSO | |
|---|---|---|---|---|---|---|---|---|
| | | | Avg. Obj. | Avg. # of function evaluations | Avg. Obj. | Avg. # of function evaluations | Avg. Obj. | Avg. # of function evaluations |
| I | m=3, n=9 | small | 469.8 | 1164.1 | 467.1 | 90.9 | 493 | 4980.4 |
| | m=3, n=9 | big | 587 | 1675.9 | 573.9 | 206.5 | 615.6 | 5698.8 |
| | m=4, n=12 | small | 722.2 | 729.7 | 719.3 | 408.5 | 778.5 | 8299.7 |
| | m=4, n=12 | big | 718.8 | 1141 | 714 | 443 | 828.5 | 8251.2 |
| | m=5, n=15 | small | 869.5 | 1940.5 | 864.1 | 84 | 973.1 | 9136.2 |
| | m=5, n=15 | big | 954.1 | 2580.3 | 948.4 | 168.4 | 1113 | 9233.2 |
| | m=6, n=18 | small | 973.3 | 1939.1 | 972.3 | 660.4 | 1135.2 | 9751.4 |
| | m=6, n=18 | big | 1072.5 | 2571.9 | 1061.1 | 692.9 | 1311.3 | 9817.8 |
| | m=7, n=21 | small | 1286.3 | 2341.6 | 1284.5 | 520.2 | 1551.9 | 10000 |
| | m=7, n=21 | big | 1201.8 | 3760 | 1200 | 971 | 1544.5 | 10000 |
| | m=8, n=24 | small | 1426.3 | 3413.7 | 1425.5 | 821.9 | 1819.7 | 10000 |
| | m=8, n=24 | big | 1411.4 | 3626.3 | 1402 | 1618.3 | 1872.7 | 10000 |
| II | m=6, n=30 | small | 1221.7 | 2174.8 | 1218.6 | 1571.3 | 2452 | 10000 |
| | m=6, n=30 | big | 1294 | 2718 | 1286.2 | 1378.6 | 2544.7 | 10000 |
| | m=7, n=40 | small | 2841.4 | 4365.8 | 2823.7 | 2068.5 | 5772.6 | 10000 |
| | m=7, n=40 | big | 2251.5 | 4421.5 | 2199.2 | 2764.2 | 4937 | 10000 |
| | m=8, n=50 | small | 2300.6 | 4452.2 | 2294.2 | 2261.6 | 6744 | 10000 |
| | m=8, n=50 | big | 3247.1 | 4491 | 3124.4 | 3678.3 | 7960.5 | 10000 |

Parameters of PSO are set as: chromosome size = 20; maximum iterations = 500; $C_1 = 2$; $C_2 = 2$; $W = 0.5$.

Table 4.9: A comparison between LF-BAP, GA and PSO on Set III

| Set | Problem size | Tidal effect | LF-BAP | | GA | | PSO | |
|---|---|---|---|---|---|---|---|---|
| | | | Avg. Obj. | Avg. # of function evaluations | Avg. Obj. | Avg. # of function evaluations | Avg. Obj. | Avg. # of function evaluations |
| III | m=9, n=60 | small | 5847.4 | 4502.5 | 5840.5 | 3034.1 | 9816.1 | 10000 |
| | m=9, n=60 | big | 5798.8 | 4406.3 | 5733.5 | 3749.2 | 9779.7 | 10000 |
| | m=10, n=70 | small | 7473.8 | 4201.2 | 7403.2 | 3565 | 13502.3 | 10000 |
| | m=10, n=70 | big | 6957.4 | 4318.9 | 6952.5 | 3483.2 | 12555.5 | 10000 |
| | m=12, n=80 | small | 7911.6 | 4273.5 | 7881.2 | 3624.2 | 13620.9 | 10000 |
| | m=12, n=80 | big | 7901.9 | 4733.9 | 7866.5 | 3371.7 | 14025.8 | 10000 |
| | m=13, n=100 | small | 10599 | 4500.7 | 10588.6 | 4042.8 | 20189.5 | 10000 |
| | m=13, n=100 | big | 10316.7 | 4192.5 | 10283.1 | 3554.6 | 19568.9 | 10000 |
| | m=14, n=120 | small | 13835.1 | 4123.8 | 13794.7 | 3474.1 | 27267 | 10000 |
| | m=14, n=120 | big | 14421 | 4332.1 | 14355.9 | 3662.1 | 27681.9 | 10000 |
| | m=15, n=150 | small | 19904.8 | 4238.3 | 19908.8 | 3780.3 | 41023.3 | 10000 |
| | m=15, n=150 | big | 20215.8 | 4579.3 | 20156.9 | 4242.8 | 40603.5 | 10000 |
| | m=20, n=200 | small | 27116.1 | 4063.3 | 27182 | 4038.6 | 58855.1 | 10000 |
| | m=20, n=200 | big | 26182.2 | 4388.1 | 26213 | 3835 | 54993.8 | 10000 |
| | m=30, n=300 | small | 39529.1 | 4117.9 | 40009.5 | 612.5 | 89470 | 10000 |
| | m=30, n=300 | big | 40281.8 | 3701.9 | 40509.2 | 626.5 | 90332.2 | 10000 |
| | m=40, n=400 | small | 52936.2 | 3817.9 | 53665.2 | 280.2 | 127400.7 | 10000 |
| | m=40, n=400 | big | 51751.4 | 3748.4 | 52320.9 | 283.5 | 122748.2 | 10000 |
| | m=50, n=500 | small | 66371.4 | 3049.6 | 67736.8 | 125.1 | 158797.7 | 10000 |
| | m=50, n=500 | big | 65460.2 | 3375.5 | 66504.7 | 169.4 | 157554.99 | 10000 |

Table 4.10: A comparison between LF-BAP, GA and PSO on Set IV

| Set | Problem size | Tidal effect | LF-BAP | | GA | | PSO | |
|---|---|---|---|---|---|---|---|---|
| | | | Avg. Obj. | Avg. # of function evaluations | Avg. Obj. | Avg. # of function evaluations | Avg. Obj. | Avg. # of function evaluations |
| IV | m=5, n=80 | small | 18295.3 | 4427.3 | 18082.9 | 3086.3 | 30819.46 | 10000 |
| | m=5, n=80 | big | 19465.6 | 4590.3 | 19012.1 | 3736.4 | 31039.91 | 10000 |
| | m=6, n=100 | small | 22131.7 | 4453.4 | 21869.2 | 3344.4 | 40558.98 | 10000 |
| | m=6, n=100 | big | 24464.4 | 4642.2 | 23929.2 | 3328.6 | 41507.12 | 10000 |
| | m=7, n=150 | small | 41710.2 | 4235.9 | 41596.7 | 3761.4 | 80416.00 | 10000 |
| | m=7, n=150 | big | 44626.2 | 4328.2 | 44122.4 | 2991.4 | 81033.95 | 10000 |
| | m=8, n=200 | small | 68726.4 | 4659.1 | 68210.9 | 3151.6 | 140833.37 | 10000 |
| | m=8, n=200 | big | 69696.7 | 4363.2 | 68420.9 | 2804.7 | 133343.53 | 10000 |
| | m=9, n=250 | small | 98179.3 | 4295 | 97847.2 | 3176.6 | 206834.05 | 10000 |
| | m=9, n=250 | big | 95080.4 | 4462.2 | 94008.2 | 2948.3 | 187646.04 | 10000 |
| | m=10, n=300 | small | 122812.5 | 4449.8 | 122557.6 | 1530.2 | 270415.29 | 10000 |
| | m=10, n=300 | big | 124669 | 4315.8 | 123241.5 | 2540.1 | 251057.32 | 10000 |

## 4.4 Conclusion

This chapter has made contributions as follows:

1. A new GA has been proposed to solve BAPs while considering multiple tides with a new neighbourhood structure to enhance the performance.
2. The efficiency of operators in GA was studied.
3. The results of GA were compared with CPLEX-BAP and H-BAP.
    - GA was faster than CPLEX in all instances. It also performed well in 73.8% of them in terms of solution quality and feasibility.
    - GA was capable of finding feasible solutions for all test cases while CPLEX model was able to solve about 68% of them.
    - GA outperformed H-BAP in all test cases.
4. The performance of meta-heuristics on BAPs was assessed by conducting experiments of LF-BAP, GA and a PSO. PSO was modified to fit BAPs with multi-tidal windows.
    - LF-BAP and GA both outperformed PSO and H-BAP in all test cases.
    - GA was able to converge more quickly and obtain better-quality solutions in most of the cases compared to LF-BAP.

# Chapter 5

# A framework of discrete event simulation

## 5.1 Introduction

Real-world optimisation problems are normally complex which are subject to uncertainty and unknown changes. As changes occur in an optimisation problem, the algorithm needs to react to the changes to produce a new optimal solution in regards to the changes. Due to the variety of uncertainty in real-world problems, it is important to effectively test an algorithm and evaluate the performance under different scenarios.

As mentioned in the literature review in Chapter 2, logistics problems at ports and container terminals face a lack of comparisons of optimisation algorithms in real-world scenarios. The performance measure varies a lot due to the number of problem variants such as BAPs with different problem settings. With different performance measures, it is difficult to compare existing methods and assess their performance. It also makes applying existing algorithms to new problems impossible. Similar problems sometimes might be solved with the same approach with small changes. Due to the diverse performance measures, without a deep analysis and experiment, researchers hardly know which approach improves a certain problem the most. For researchers lacking computer science background, it is too challenging to see the flow in complex or dynamic optimisation problems. With visualisation, complex information can be simplified to 2D graphs or 3D animations. It is a critical component to understand the behaviour of simulation models and help researchers to improve the model.

Therefore, it is necessary and important to fill the gap. In order to fill the gap there are several goals we need to achieve: 1) being able to generate test cases for different problems; 2) easy integrating with optimisation algorithms; 3) applying for benchmarking; 4) identifying common performance measures; 5) the ability to visualise.

In this chapter, a new framework of discrete event simulation is proposed as a tool to compare and visualise logistic optimisation problems at ports. Meanwhile, it is also used to generate and test instances of optimisation problems in different scenarios. This framework is developed based on an open-source simulation software named JaamSim. To the best of our knowledge, this is the first time a discrete-event simulation is used as a flexible framework for different logistic problems. The framework is proposed as a complex of optimisation and simulation. As mentioned in Section 2.2, discrete event simulation is one of the most popular models in simulating port operations. First, the event-based nature of discrete-event simulation makes it possible to generate different problem instances very easily by just creating different events and adjusting how the system changes its state upon an event. Second, generating dynamics is the nature of simulation - all simulation software is intrinsically equipped with some random/uncertainty generator. This makes simulations naturally suitable for generating problem instances with uncertainty. Third, the visualisation feature of simulation can help researchers visually observe the behaviour of algorithms much more easily. Fourth, many simulation software packages have drag-and-drop features, which would make the process of creating logistics problems more user-friendly. Fifth, discrete-event simulation is naturally suitable for complex systems. This makes it suitable for creating more challenging optimisation problems. Finally, the flexibility of open source makes it possible to easily integrate different algorithms and extend the framework to other type of problems.

In the following section, we start from a description of the proposed framework including the structure and the software for development. Then examples of using the simulation framework are shown in Sections 5.3 and 5.4. Bin packing problems (BPPs) and BAPs are modelled as demonstrations. The contributions are summarised in Section 5.5.

## 5.2 The framework description

This section explains the framework by firstly introducing the software used as a developing tool and then explaining the structure of the framework. It then introduces the features of this framework and how to integrate it with optimisation algorithms.

### 5.2.1 Simulator in the framework

There are a variety of simulation software and libraries in the market including commercial software and open source software. According to the survey about simulation software used at ports [Dragović et al., 2017], commercial simulation software such as Arena and FlexSim, provide powerful 3D visualisations and a large number of individual items for users to build a model. The most used software in this field according to the literature is Arena. However, many of them have only been used once in the literature. In general, the development in commercial software is restricted to functions pre-built. It means users are only allowed to make changes to specific features.

Thanks to access to the source code in open source simulation software like SimPy and JaamSim, users are provided more flexibilities to address different features in their problem. Some works of open source simulation software are reviewed in Dagkakis and Heavey [2015]. Among them, JaamSim written in Java shows promising development and active forum [King and Harrison, 2013]. Key functions of JaamSim include drag-and-drop interface, basic objects for process flow models, impressive 3D visualisation, controls for launching and manipulating simulation runs and model editors. It also provides probability distributions for random sampling and dynamic graphs for analysis. More importantly, as an open source software, most of the functions and objects can be modified based on developers needs. For example, developers are able to create their own high-level objects. They can also customise any event or simulation process by accessing the source code. The whole system is faster than much commercial software in terms of event processing and execution time of entities [King, 2014].

Therefore, we believe JaamSim is a perfect fit in operating this framework. JaamSim is an important part in executing the simulation process and the optimisation algorithm. All the operations will be done in JaamSim through its interface and all the results and performance will be displayed. It asks users to choose a problem including objective

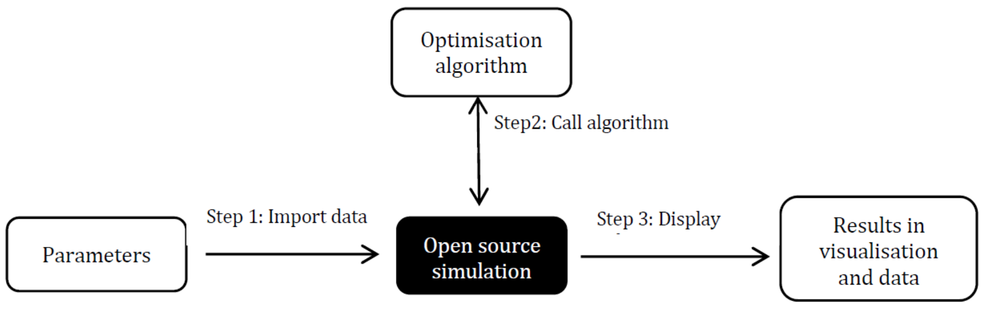


FIGURE 5.1: Framework structure.
Optimisation problems will be simulated using the open source simulation engine; parameters are imported as input in step 1 so that test cases are generated. In step 2, while running the simulation, an algorithm is called from the open-source simulation engine and the packing result for each item is returned to the engine. Then the result is displayed in step 3 as output.

and all the other necessary parameters such as a distribution for an uncertainty. It will also ask users to choose the directory to output the test case and the location of the algorithm if testing the performance of an algorithm. The framework is built by using drag-and-drop as well as working on the source code. The architecture of the framework is given in Fig. 5.1.

### 5.2.2 Instructions of the framework

We propose this framework with following uses. Firstly, the framework supports simulations of numerous optimisation problems. JaamSim provides basic objects which have broad applications, such as queues and servers. In order to build a complex model, it also allows users to create new pallets of high-level objects. Therefore, with this simulator the framework is able to adapt models like manufacturing, traffic control and scheduling etc. Moreover, a solution of comparing optimisation algorithms and exploring improvements is offered by proposing this framework. The framework helps researchers evaluating their algorithms in different environments and it also benefits the industry by assessing the performance of different optimisation methods. In the meantime, there is a function of generating customised test instances aiming to evaluate the performance under different scenarios. By providing necessary inputs, the difficulty of the scenario and further stochastic elements are defined.

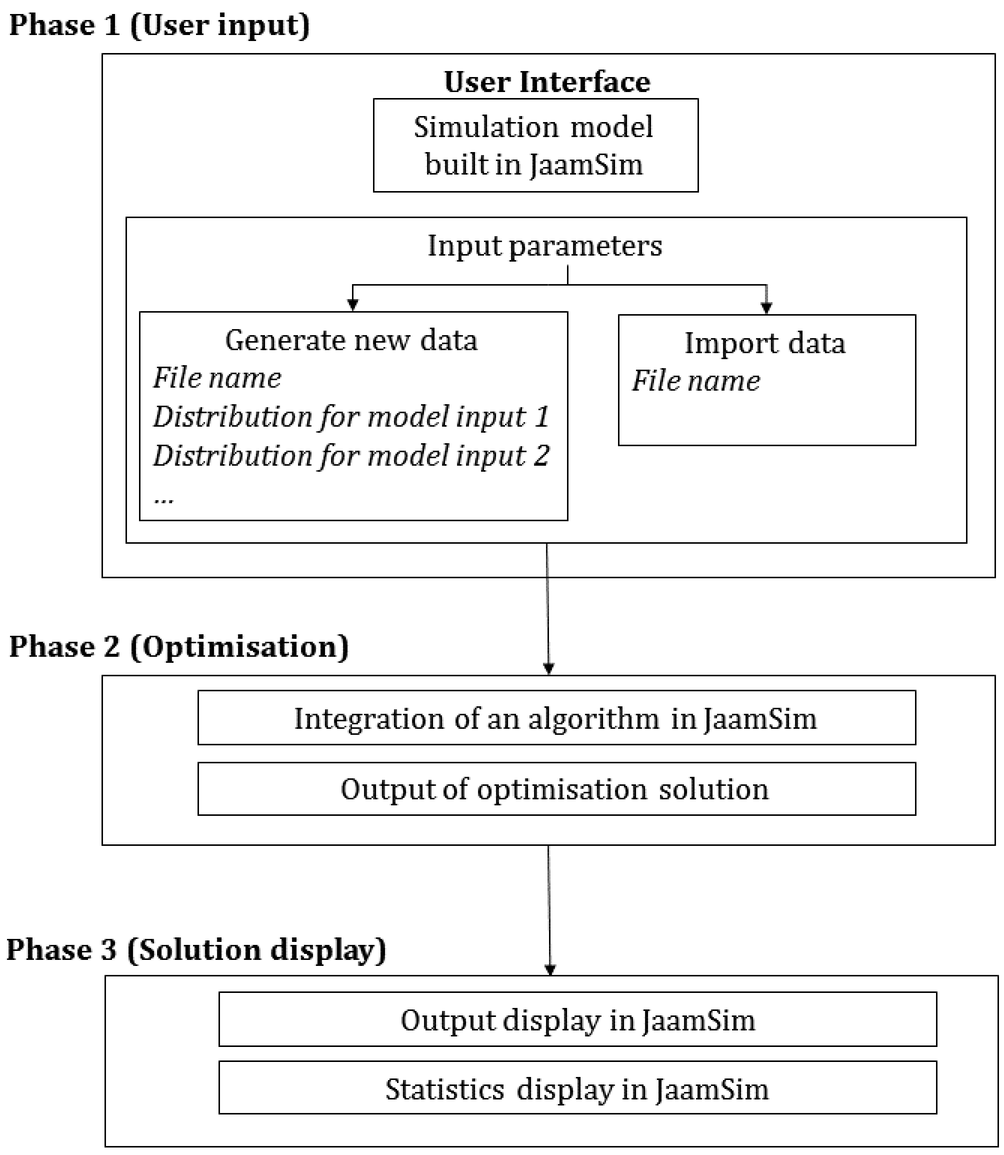

Phase 1 (User input)
User Interface
Simulation model built in JaamSim
Input parameters
Generate new data
File name
Distribution for model input 1
Distribution for model input 2
...
Import data
File name
Phase 2 (Optimisation)
Integration of an algorithm in JaamSim
Output of optimisation solution
Phase 3 (Solution display)
Output display in JaamSim
Statistics display in JaamSim

FIGURE 5.2: Flow chart of the framework

As explain in Section 5.1, the simulation framework should support different optimisation problems. To test the performance of an optimisation algorithm with this framework, there are several steps to follow. The process is displayed as a flow chart in Fig. 5.2.

**User input** In Phase 1, with the user interface in JaamSim, the simulation model is built. In order to decide the specific constraints and the objective function for the problem, a number of problem variants are pre-defined. Then users need to input parameters if generating new data instance. By choosing certain parameter for each criteria such as a distribution, users customise the scenario and the test instances are generated. If using existing data, it can be imported by providing the directory. For now the input of parameters can only be defined through the source code. In the future, a user interface should be developed in order to allow a number of selections.

**Optimisation** Once the optimisation algorithm is integrated in the simulation engine, the model executes it as an external function. It reads the existing data or the test instance generated in Phase 1, along with all the other necessary information in the simulation model. After running the algorithm, the solution is achieved with local output and the optimisation plan is sent back to the simulation model.

**Solution display** The simulation model will automatically collect the new information and update the operations. Meanwhile, the visualisation of the whole process also shows how the problem is optimised and how the algorithm performs under this scenario. For the purpose of comparing different algorithms and test them under numerous scenarios, the simulator also shows the customised statistics.

## 5.3 A case study of Bin packing problems

In this section, a three-dimensional (3D) bin packing problem (BPP) is simulated as an example of using the proposed framework. Firstly, objectives and constraints are explained for this problem. Then the choices of algorithms are introduced. With the stated format of input, the simulation process is summarised and the experiment is conducted.

3D BPPs have drawn a lot of attention from the industry because of the difficulty in solving them in real-world scenarios. BPPs are applied as a demonstration of using this framework in this section. 3D BPPs are NP-hard optimisation problems in which a number of boxes are packed into one or multiple 3D bins. Depending on the characteristics of the problem, different objectives can be defined for BPPs such as input minimisation or output maximisation. Input minimisation aims to find the minimum total cost or minimum number of bins. The size of bins can be either identical [Crainic et al., 2009, Feng, 2013, Martello, 2007] or varied [de Almeida, 2010, Alvarez-Valdés et al., 2013, Che, 2011, Eley, 2003, Tian et al., 2015]. The aim of output maximisation is maximising the volume or number of packed boxes given the limited number of bins [Lim, 2013, Liu, 2011, Junqueira, 2012, Costa and Captivo, 2016]. Some BPPs have multiple objectives:

minimising the cost while packing items with preferences [Tian et al., 2015]; minimising the cost and also packing items with the same destinations together [Ceschia and Schaerf, 2013]. In terms of size, items may be identical, weakly heterogeneous (i.e. many items but a few item types), or strongly heterogeneous (i.e. a few items but many item types). Regarding constraints on BPPs, Bortfeldt [Bortfeldt, 2012] introduced various constraints on potential containers, items, loading and allocation.

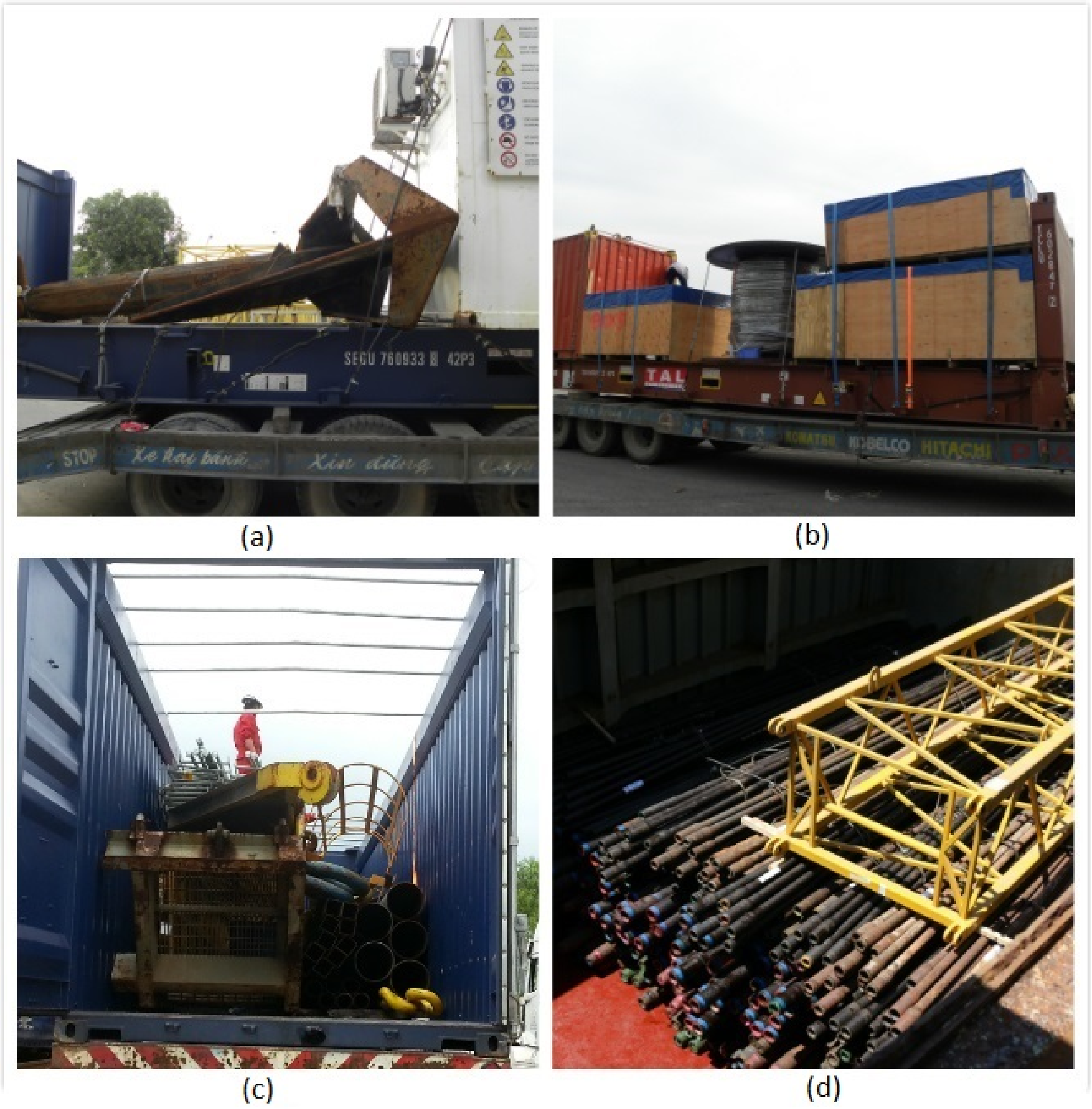


FIGURE 5.3: Examples of some objects in the real-world case study Bin packing problems.
(a) An anchor. No other item can be placed on top of this anchor. (b) A cable organiser and some rectangular boxes placed in an open-top container. (c) and (d) How non-rectangular items are stacked on each other and how tubes are bundled.

In the literature the BPPs are normally considered in an ideal situation with no uncertainty. In reality, however, uncertainty is a frequent feature of real-world BPPs. A case from our industrial partner is explained here as an example: a multiple bin-size Bin packing problem. Based on the data provided by an industrial partner, there is a set of items with different sizes to be packed into containers whose sizes (20, 40, 45 feet) and structures (closed, open or flat rack) depend on the sizes of the items. Items include boxes, anchors, tubes, and some other odd shape items fixed on a palette (Fig. 5.3). The cost of hiring containers varies depending on their sizes and structures. Assuming the number of containers is unlimited, the objective is to minimise the total hiring cost of containers in order to pack all items needed. In summary, the constraints for this problem are as follows: 1) items can be rotated but not up-side-down; 2) all items must be placed parallel to the containers sides; 3) the total weight of items packed in each container must not exceed the weight that the container could bear and or the weight limit of the carrying vehicle; 4) items may not intersect each other and some items cannot be stacked on top of another. This problem, similar to other real-world optimisation problems, is subject to uncertainty. For example, some items can be tied together into a bundle. How many items, and how they can be tied together into one bundle, is uncertain. The size of each bundle is also uncertain, depending on how items are tied together.

Hence, we need to take uncertainty into account in academic research. To address uncertainty to a variety of BPPs, it is necessary to have a data set for each problem based on different constraints and features of the problem. In following sections, We explain the features of this framework in solving BPPs and the implementations.

### Objectives and uncertainty

The framework provides three objectives for BPPs that optimisation algorithms can choose to optimise: 1) minimise the number of bins; 2) minimise the cost of bins; 3) maximise the profit of packed items.

Given the objectives above, the framework can generate uncertainty in: size of items, weight of items, profit of items, and cost of bins. The uncertain values can be generated under any distribution (e.g. uniform distribution, normal distribution). Both two-dimensional (2D) and 3D (cuboid items) BPP can be visualised. The framework can generate problems with single bin, multiple identical bins, or multiple bins of different type.

### 5.3.1 Simulation process

The simulation process for BPPs is outlined as follows.

1. Setting up inputs to generate test cases. The format of input for minimising the number of identical bins is shown in Table 5.1. The details of generating test cases and examples are explained in Section 5.3.2.

2. Once a test instance is generated, the algorithm is called while passing the information of this item as parameters (width, height, length). The test cases are saved as a text file as well in the format below.

   $n$ $b_x, b_y, b_z$

   $a_{1,x}$ $a_{1,y}$ $a_{1,z}$

   $a_{2,x}$ $a_{2,y}$ $a_{2,z}$

   ...

   ...

   ...

   $a_{n,x}$ $a_{n,y}$ $a_{n,z}$ where $n$ is the total number of items, $x, y, z$ are the width, height and length.

3. Then the packing location is returned by the algorithm so that the framework could visualise.

4. In the meanwhile, the packing result is exported to a file with a format as follows: bin type, bin no., item width, item height, item length, packing position coordinate x, y, z.

### 5.3.2 Generating test problems

This subsection proposes test cases that we suggest to use as default. However, our framework is not limited to them. Users can customise test cases based on their own needs. Here we considered three levels of test cases: easy, medium, and hard which represent the level of difficulty of packing items. The time that an algorithm takes normally depends on the difficulty of the problem. Due to the lack of a proper 3D benchmark, the following instances are extended from Lodi et al. [Lodi et al., 1999]

Table 5.1: The format of input for bin number minimisation

| Parameters | Explanation |
|---|---|
| String file_name | The output file name of test cases. It should be the same name read in algorithm. |
| int bin_x, int bin_y, int bin_z | The size of bins. |
| int item_num | The total number of items to be packed. |
| int x_dis, int y_dis, int z_dis | The distribution of each dimension (width, height, length). 1 is normal distribution, the default is uniform distribution. |
| int x_min, int y_min, int z_min | It is the mean for normal distribution or the lower bound of the range for uniform distribution. |
| int x_max, int y_max, int z_max | It is the standard deviation for normal distribution or the upper bound of the range for uniform distribution. |

which provides instances in 2D. Another dimension is added for bins and items. They are displayed in Table 5.2 based on the following types of cuboids that are defined in terms of the width W, height H and length L of the bins.

Type 1: $w_j$ uniformly random in [$\frac{2}{3}$W, W]; $h_j$ uniformly random in [1, $\frac{1}{2}$H]; $l_j$ uniformly random in [1, $\frac{1}{3}$L];

Type 2: $w_j$ uniformly random in [1, $\frac{1}{2}$W]; $h_j$ uniformly random in [$\frac{2}{3}$H, H]; $l_j$ uniformly random in [$\frac{1}{2}$L, L];

Type 3: $w_j$ uniformly random in [$\frac{1}{2}$W, W]; $h_j$ uniformly random in [$\frac{1}{2}$H, H]; $l_j$ uniformly random in [1, $\frac{1}{2}$L];

Type 4: $w_j$ uniformly random in [1, $\frac{1}{2}$W]; $h_j$ uniformly random in [ 1, $\frac{1}{2}$H ]; $l_j$ uniformly random in [$\frac{2}{3}$L, L].

The input used for generating the data set is listed below. The fifth parameter, n is the number of instances which must be an integer. In Java, the specific function with the same name will be called depending on the number of input parameters. We have two combinations of parameters below. The first one generates instances by choosing a specific distribution (data set I, II, III), and the instances generated by the second one are based on the size of bins (data set IV). The n below represents the number of items to be generated.

**Parameters** String file_name, int bin_x, int bin_y, int bin_z, int item_num, int x_dis, int x_min, int x_max, int y_dis, int y_min, int y_max, int z_dis, int z_min, int z_max

OR

String file_name, int bin_x, int bin_y, int bin_z, int item_num

**I:** "test_cases1.txt", 30, 30, 30, n, 0, 1, 10, 0, 1, 10, 0, 1, 10

**II:** "test_cases2.txt", 100, 100, 100, n, 0, 1, 35, 0, 1, 35, 0, 1, 35

**III:** "test_cases3.txt", 100, 100, 100, n, 0, 1, 100, 0, 1, 100, 0, 1, 100

**IV:** "test_cases4.txt", 100, 100, 100, n

### 5.3.3 Algorithms integration

The framework supports solving the BPP in both static (offline) and dynamic (online) ways. In the static case, all items are available beforehand, and the optimisation algorithm can freely choose any item to load into a bin. In the dynamic case, the bins need to be packed when time goes by, and items arrive at different times. Whenever one or some item(s) arrive, the algorithm needs to find the best way to pack the items into a bin, then waits for the next set of items to come, and so on.

To illustrate how the framework can be used to test algorithms that solve the static BPP, we use a static bin packing algorithm [Martello, 2007] which solves the BPP in an offline way, assuming that all items are available beforehand. This algorithm uses heuristic approaches on initially sorted items by non-increasing volume. The algorithm also assumes that unlimited identical bins are given, and bins have fixed orientation. If one bin is full then it is closed, a new bin is set as open to receive items. The algorithm was written in C, then was compiled into an executable file, which is then called by the simulation framework. We choose this algorithm because, although it is not the latest method, it is one of the few available 3D bin packing algorithms whose source code is accessible and a detailed algorithm description is available. Because the purpose of this section is to provide a proof of concept, we feel the decision of choosing this algorithm is justified.

In addition, we develop a new online algorithm to illustrate how the framework can be used to test the dynamic BPP in Section 5.3.3.1.

Table 5.2: Example of the problem instances generated by the framework

| Data set No. | Category | Bin size (W*H*L) | Total number of items | Item ($w_j$*$h_j$*$l_j$) |
|---|---|---|---|---|
| I_20 | Easy | 30*30*30 | 20 | Uniformly random in [1, 10] |
| I_40 | Easy | 30*30*30 | 40 | Uniformly random in [1, 10] |
| I_60 | Easy | 30*30*30 | 60 | Uniformly random in [1, 10] |
| I_80 | Hard | 30*30*30 | 80 | Uniformly random in [1, 10] |
| I_1000 | Hard | 30*30*30 | 1000 | Uniformly random in [1, 10] |
| II_20 | Easy | 100*100*100 | 20 | Uniformly random in [1, 35] |
| II_40 | Medium | 100*100*100 | 40 | Uniformly random in [1, 35] |
| II_60 | Medium | 100*100*100 | 60 | Uniformly random in [1, 35] |
| II_80 | Hard | 100*100*100 | 80 | Uniformly random in [1, 35] |
| II_1000 | Hard | 100*100*100 | 1000 | Uniformly random in [1, 35] |
| III_20 | Medium | 100*100*100 | 20 | Uniformly random in [1, 100] |
| III_40 | Medium | 100*100*100 | 40 | Uniformly random in [1, 100] |
| III_60 | Hard | 100*100*100 | 60 | Uniformly random in [1, 100] |
| III_80 | Hard | 100*100*100 | 80 | Uniformly random in [1, 100] |
| III_1000 | Hard | 100*100*100 | 1000 | Uniformly random in [1, 100] |
| IV_40 | Hard | 100*100*100 | 40 | Type 1 with probability 70%, Type 2, 3, 4 with probability 10% each |
| IV_1000 | Hard | 100*100*100 | 1000 | Type 1 with probability 70%, Type 2, 3, 4 with probability 10% each |

#### 5.3.3.1 Developing a new online algorithm for 3D BPPs

For the purpose of benchmarking dynamic 3D BPPs, we develop a new online algorithm (Algorithm 11). How the framework can be used to test dynamic algorithms is also illustrated. It should be noted that the dynamic BPP is very new to the academic community, and while there have been a few research studies that have proposed solving algorithms [Burcea et al., 2013, Epstein and Levy, 2010, Wong and Yung, 2010], these research studies have not provided any experimental details to prove that these algorithms work. Because of that, here we just provide a simple algorithm as a proof of concept. The algorithm is written in Java, and it works by packing upcoming items layer by layer under the same assumption as the static algorithm. In addition, the algorithm assumes that there is no information of upcoming items and hence the problem needs to be solved online. At the time an item comes, the information of the item, the current bin and a packing location are passed to the algorithm as parameters and the algorithm goes on packing the new item into the currently available bins. Items are packed in a bin in "layers" (see Figure 5.4 and Figure 5.5).

The size of each layer is the maximum size of the packed items. The algorithm will check whether the current vertical layer in a bin has enough space for this item. If there is enough space on the current layer, the item is packed to the location and returns the updated packing location. If there is not enough space on this layer, it will check the next layer till this bin is full and then open a new one.

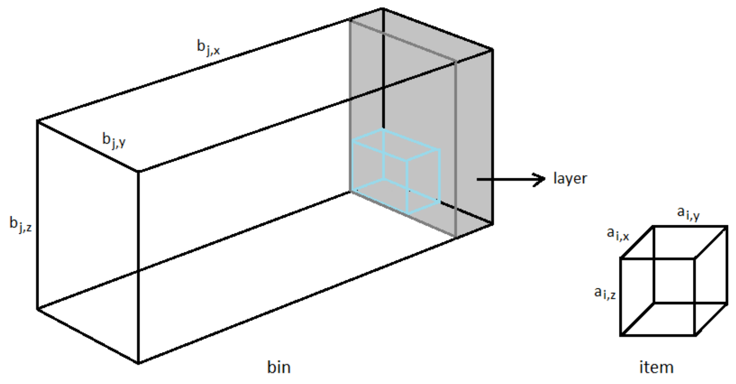


FIGURE 5.4: Example of item, bin, and layer in 3D

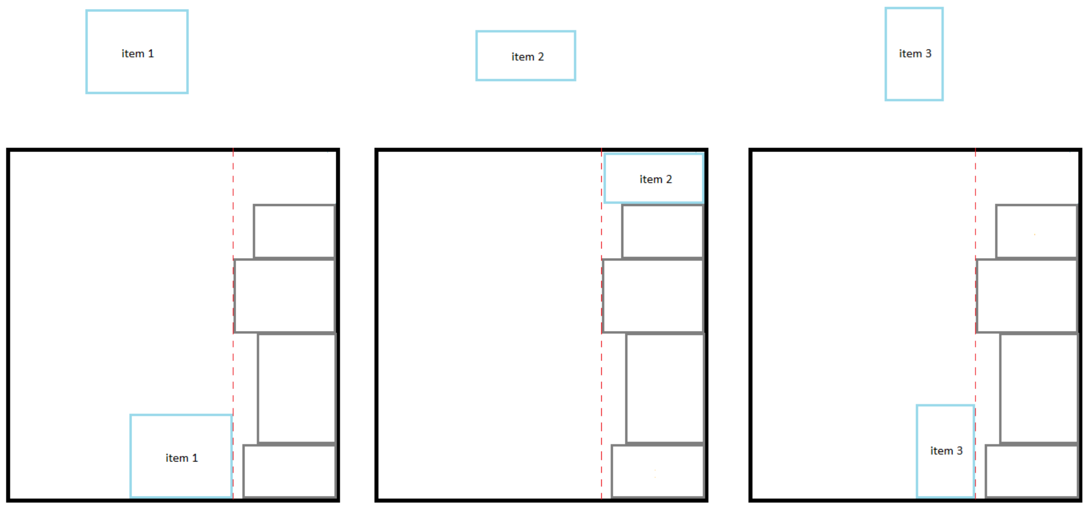


FIGURE 5.5: Example of packing in a layer in 2D using Algorithm 11.
Four items are packed in the first column. For the newly arrived item 1, the space left in the first column is not enough, so it is placed to the second column. Because the volume of item 2 is smaller than the free space, it is placed on the top of those items in the first column. Item 3 has the same volume as item 2 but with different orientation. It is placed to the second column since our algorithm is under the assumption of fixed orientation.

**Algorithm 11** OnlineBinPacking($a_i$, $b$, $p$)

```
**While** $j^{th}$ bin exists **do**
    **If** $a_{i,x} > b_{j,x} || a_{i,y} > b_{j,y} || a_{i,z} > b_{j,z}$    //if any of the dimensions of the item is larger than the bin
        //this item can not be packed
        **Return**
    **If** $p_x + a_{i,x} > b_{j,x}$    //if the space through x coordinate is not enough
        //move the packing location to the origin of next bin, close the current bin
        update $p$ and $j$ to next bin
    **Else if** $p_y + a_{i,y} > b_{j,y}$    //if the space through y coordinate is not enough
        //-move the packing location to the next layer
        update $p$ to next layer through x coordinate
    **Else if** $p_z + a_{i,z} > b_{j,z}$    //if the space through z coordinate is not enough
        //-move the packing location to the next column
        update $p$ to next layer through y coordinate
    **Else do** $p_z + a_{i,z} > b_{j,z}$    //space is enough, pack the item
        //pack the item and update $p$, $i$ and $j$
        update $p$ to next layer through y coordinate
        **Return**
**Return**
```

where $p_x$,$p_y$,$p_z$ are the packing location, $a_{i,x}, a_{i,y}, a_{i,z}$are the width, height and length of the $i^{th}$ item in the list of items, $b_{j,x}, b_{j,y}, b_{j,z}$are the width, height and length of the $j^{th}$ bin in the list of bins.

### 5.3.4 Experiment

The optimal solution that a dynamic BPP algorithm could find in the online dynamic case would always be worse than or, at best, equal to the optimal solution found in the static case. Due to that, to evaluate the efficiency of a dynamic BPP algorithm, we can compare its solution with that of an established static BPP algorithm.

To demonstrate this type of comparison, in this experiment we are going to compare our online algorithm (Algorithm 11 in Section 5.3.3.1) with the static algorithm from Martello [2007] to evaluate the effectiveness/efficiency of Algorithm 11. The test set in Table 5.2 is applied to both algorithms. Different numbers of items are set for each group of instances. In each group, we run the algorithm for ten replications.

**Performance measures**

As mentioned in Chapter 2, performance measures are necessary to be introduced due to the lack of common measures. To evaluate the effectiveness of the algorithms compared to other algorithms, we use the following measures: average utilisation and number of bins. The average utilisation shows how much the bin capacity is used on average. It provides a reference to see how close the items are packed in each bin. The average utilisation of the bins is the total volume of packed items divided by the total volume of used bins. Moreover, the number of bins is used as another performance measure regarding the effectiveness of algorithms. It represents how an algorithm performs with an objective of input minimisation such as number of bins minimisation or cost of bins minimisation.

Performance measures regarding the efficiency of algorithms generally means the process time of an algorithm, for example, how fast or slow the algorithm identified the optimum solution. The criterion that we will use in this experiment at the moment the framework supports is the running time, which is the total process time of an algorithm in our framework. This criterion can be easily collected and it is valuable in comparing performances of different algorithms on computers of the same standard. Other criteria like CPU usage could be implemented in the future. Therefore, in this experiment we use the number of bins used, the average utilisation and the running time (see Section 5.3.4).

Table 5.3: Bin packing results.

| Data set No. | Total No. of items | Online No. of bins | Online Average utilisation | Online Running time (s) | Static No. of bins | Static Average utilisation | Static Running time (s) |
|---|---|---|---|---|---|---|---|
| I_20 | 20 | 1 | 11.994% | 0.001 | 1 | 11.994% | 0.129 |
| I_40 | 40 | 1.1 | 24.054% | 0.001 | 1 | 25.752% | 0.136 |
| I_60 | 60 | 2 | 19.037% | 0.001 | 1 | 38.077% | 0.116 |
| I_80 | 80 | 5 | 37.43% | 0.001 | N/A | N/A | N/A |
| I_1000 | 1000 | 22.3 | 29.16% | 0.001 | N/A | N/A | N/A |
| II_20 | 20 | 1 | 11.619% | 0.001 | 1 | 11.619% | 0.226 |
| II_40 | 40 | 1.4 | 17.442% | 0.001 | 1 | 22.273% | 0.146 |
| II_60 | 60 | 2 | 16.92% | 0.001 | 1 | 33.83% | 0.19 |
| II_80 | 80 | 3.2 | 13.71% | 0.001 | N/A | N/A | N/A |
| II_1000 | 1000 | 26 | 22.46% | 0.002 | N/A | N/A | N/A |
| III_20 | 20 | 8.8 | 29.77% | 0.001 | 4.5 | 58.43% | 2.001 |
| III_40 | 40 | 15.7 | 29.633% | 0.001 | 6.6 | 70.107% | 14.214 |
| III_60 | 60 | 24.3 | 32.13% | 0.001 | N/A | N/A | N/A |
| III_80 | 80 | 37 | 33.11% | 0.001 | N/A | N/A | N/A |
| III_1000 | 1000 | 382 | 32.77% | 0.001 | N/A | N/A | N/A |
| IV_40 | 40 | 2 | 25.54% | 0.001 | N/A | N/A | N/A |
| IV_1000 | 1000 | 81 | 42.44% | 0.001 | N/A | N/A | N/A |

N/A represents that an algorithm has not finished the job in 10 hours.
The simulation runs on an Intel Core 2, 3.06 GHz computer with 4.0 GB RAM.

Table 5.3 shows the average results of ten replications. The average numbers of bins determined by the online and static algorithms do not have a significant difference when the number of items are low or the sizes of items are small in proportion to bin sizes, i.e. I_20 and II_20. However, for I_40, I_60, II_40, II_60, III_20 and III_40 the static solution provides a smaller number of bins with a larger average utilisation rate. This reflects the fact that in most cases the solution found online would be worse than the solution found offline in a static way.

In terms of the running time, the online algorithm shows significantly shorter process times in comparison with the static algorithm. According to the results, the static algorithm takes much longer to achieve a packing plan for a hard level of test cases. The online algorithm, on the contrary, is much faster and can solve all the problems, including the hard, large-scale instances. For example, in the results of I_1000, II_1000, III_1000 and IV_1000 shown in Table 5.3, for the online algorithm it takes only 0.002 seconds at most. This solving time includes both computational time of the proposed online algorithm and the simulation time to visually display the loading process. It means the simulation is also fast and our framework is capable of handling problems with a large number of items which is common in the real-world BPPs. Figure 5.6 shows an example of how the process of packing bins online is displayed in 3D in our proposed framework.

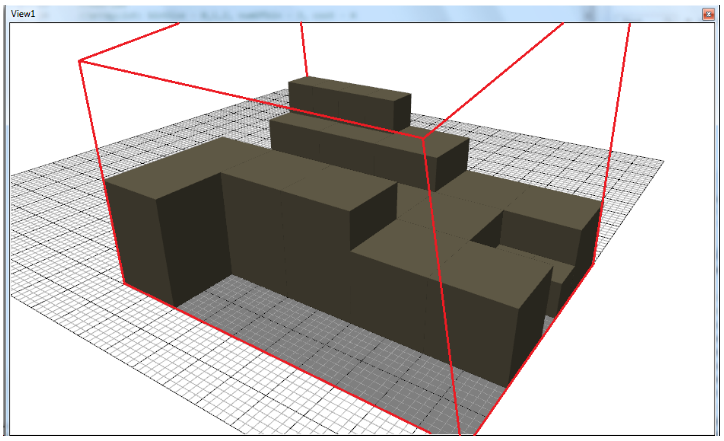


Figure 5.6: A 3D view of the online bin packing process, as displayed by our framework.

## 5.4 A case study of Berth allocation problems

### 5.4.1 Developing a simulation model of BAP

Another case study of a BAP is conducted in this section for the purpose of examining the performance of the optimisation algorithm LF-BAP proposed in Chapter 3 with 3D visualisation. We firstly explain the structure of JaamSim and then introduce the new modules developed for BAP models. The experiments with the format of the input data are described in Section 5.4.2.

The basic architecture of JaamSim has been displayed as a UML diagram in Fig. 5.7. JaamSim separates two objects *Entity* and *Process*. An active *Entity* could have multiple *Processes* going on. Processes are managed by the *EventManager* which controls the events. The *EventManager* contains an *eventStack* and a *conditionalList* in order to maintain the discrete-event logic. Both conditional events and future events of each entity are stored in event lists. Events with the same event time are sorted in order of their priorities. If certain conditions of a conditional event are satisfied, the event is executed. In order to build the model of the BAP, several new modules are developed.

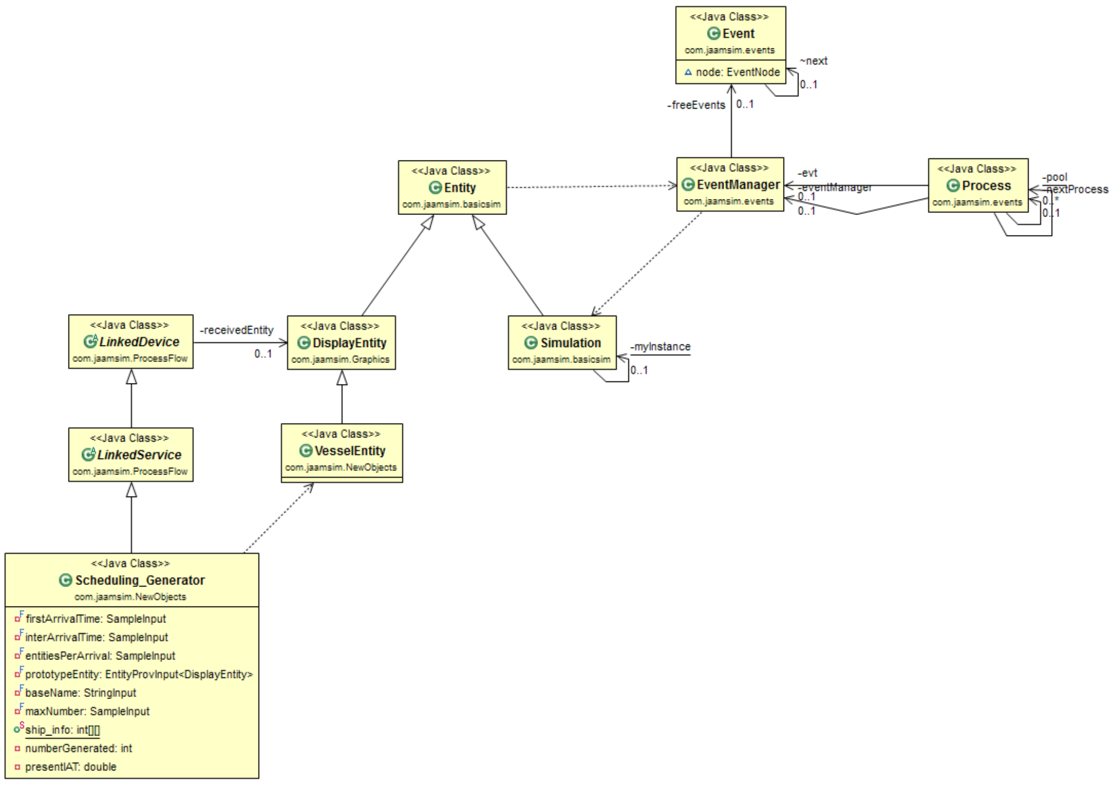


FIGURE 5.7: A UML diagram of the existing JaamSim structure with the newly defined objects

In order to integrate LF-BAP in the simulation and test on different scenarios, firstly a simulation model needs to be built by developing new modules. By following the structure of the proposed framework in Section 5.2, several characteristics of the proposed framework are shown including the simplicity of use, the 3D visualisation and potentially further comparison with other algorithms. As the prototype of the vessel object to be generated, *VesselEntity* inherits from DisplayEntity in order to support 3D graphics. By inheriting from *LinkedService*, *VesselScheduler* creates sequence of DisplayEntities at random intervals, which are placed in a target Queue. The key modules are further explained as below and a flowchart of the BAP simulation using the framework is displayed in Fig. 5.8.

### Entity module

A new object *VesselEntity* is designed to simulate and monitor vessel behaviours in the process of scheduling. Necessary information is stored as attributes, such as vessel type, cost of waiting, handling time, etc (Fig. 5.9). Different figures are defined for each type of vessel. Another key object in this simulation model is the built-in Server in JaamSim. Server is used as berth in BAP simulation.

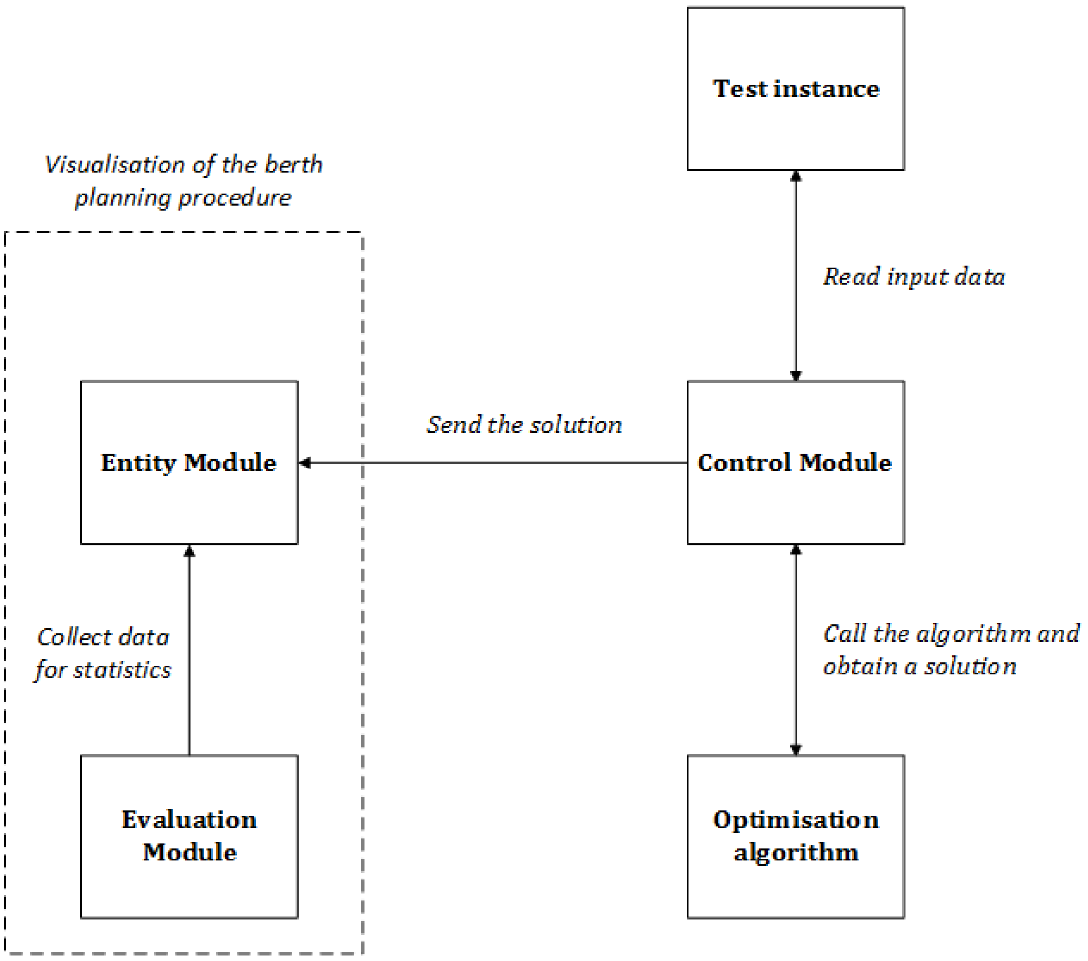
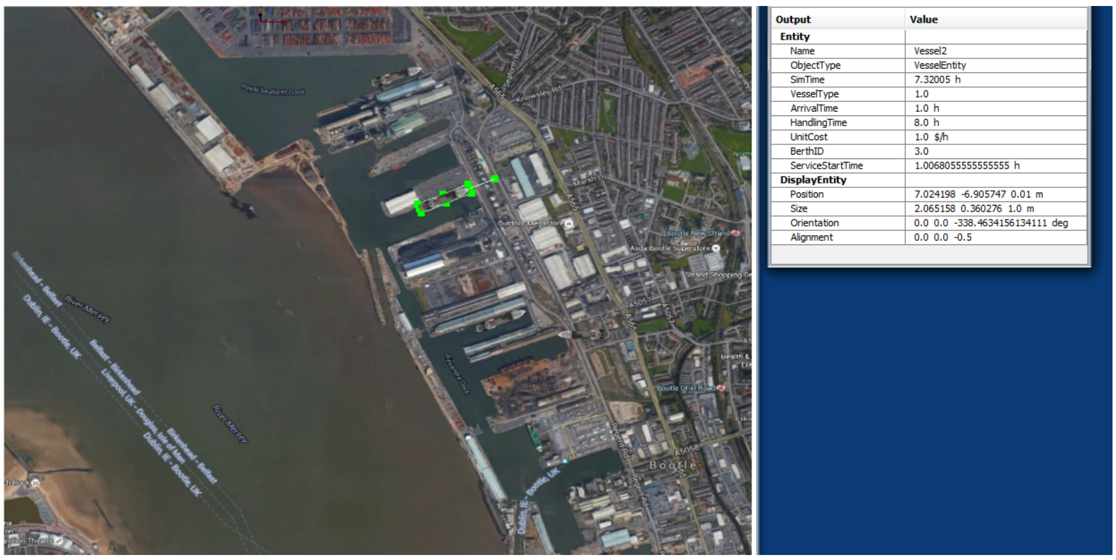

Test instance
Visualisation of the berth planning procedure
Read input data
Send the solution
Entity Module
Control Module
Collect data for statistics
Call the algorithm and obtain a solution
Evaluation Module
Optimisation algorithm

FIGURE 5.8: A flowchart of the BAP simulation using the framework.



| Output | Value |
|---|---|
| **Entity** | |
| Name | Vessel2 |
| ObjectType | VesselEntity |
| SimTime | 7.32005 h |
| VesselType | 1.0 |
| ArrivalTime | 1.0 h |
| HandlingTime | 8.0 h |
| UnitCost | 1.0 $/h |
| BerthID | 3.0 |
| ServiceStartTime | 1.0068055555555555 h |
| **DisplayEntity** | |
| Position | 7.024198 -6.905747 0.01 m |
| Size | 2.065158 0.360276 1.0 m |
| Orientation | 0.0 0.0 -338.4634156134111 deg |
| Alignment | 0.0 0.0 -0.5 |

FIGURE 5.9: Relevent information of vessels in simulation.

In the progress of the simulation, all the relevant information of each vessel can be seen in an output view.

**Control module**

*VesselScheduler* is newly proposed to import input data and execute the BAP algorithm. Once the data is imported, information of vessels is passed to vessel entities. According to the process in Fig. 5.1, after calling a BAP algorithm from *VesselScheduler*, the results (the berth allocated to each vessel) are sent to vessel entities.

**Evaluation module**

The evaluation module is used to monitor the statistics of the model. *Statistics* and *Graph* are used to show the collected statistical information.

### 5.4.2 Experiment

We conduct an experiment to show the actual schedule generated by LF-BAP as a demonstration. The format of input for simulation of BAPs with tidal windows is as below.

$n\ m\ TF$

1 $t\_proc_1\ t\_arr_1\ w_1\ L_1\ H_1$

2 $t\_proc_2\ t\_arr_2\ w_2\ L_2\ H_2$

...

...

...

n $t\_proc_n\ t\_arr_n\ w_n\ L_n\ H_n$ where $n$ is the total number of vessels, $m$ is the total number of berths, $TF$ is the tide changing frequency, $t\_proc$, $t\_arr$ and $w$ indicate the processing time, the arrival time and the unit cost of waiting time, $L$ and $H$ are the indicators of the availability of berths at low tides and high tides.

With the simulation, we can see the process of assigning each vessel as well as the customised statistics (Fig. 5.10). The simulation is able to automatically capture the critical information while running the model. We test 10 instances with 5 berths and 15 vessels. Experimental results are displayed in Table 5.4 including the total weighted cost (Eq. 3.1), average waiting time of vessels and utilisation rate of each berth.

The instances are the same as those used in Chapters 3 and 4. The variance of the results is due to different distributions used for generating data. According to the problem

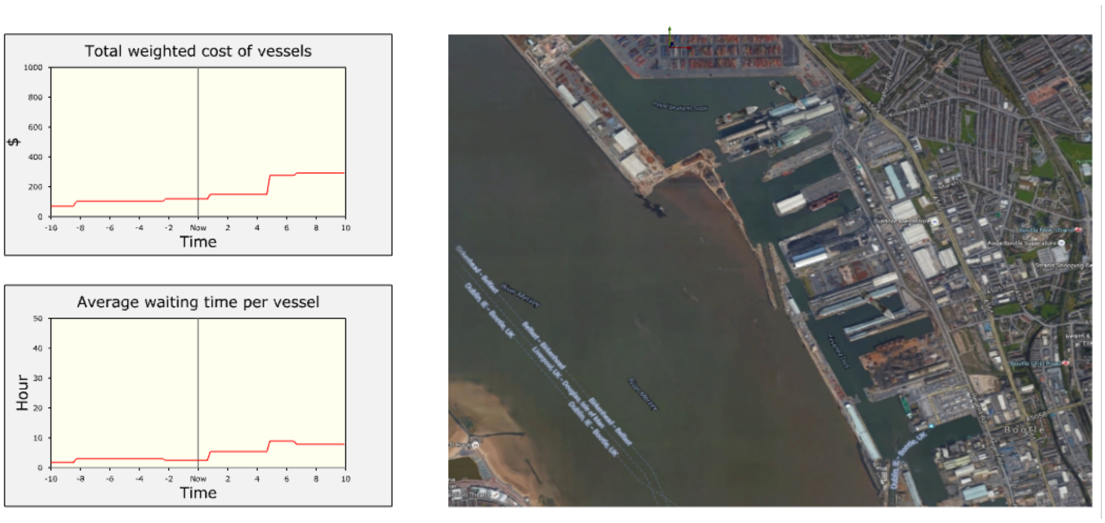

Total weighted cost of vessels
$
Time
Average waiting time per vessel
Hour
Time
Now

FIGURE 5.10: A screenshot of the statistics and the simulation.
According to the objective, the measurements are defined and collected in the simulation. The selected criteria can be displayed in real time.

TABLE 5.4: Berth allocation results.

| Test case | Total cost | Avg waiting time of vessels | Utilisation rate of berth 1 | Utilisation rate of berth 2 | Utilisation rate of berth 3 | Utilisation rate of berth 4 | Utilisation rate of berth 5 |
|---|---|---|---|---|---|---|---|
| 1 | 919.1 | 3.8h | 19.1% | 23% | 44% | 51.7% | 82.3% |
| 2 | 836 | 2.4h | 0 | 56.1% | 58.2% | 60.1% | 97.7% |
| 3 | 2088 | 14.2h | 17.6% | 16.1% | 23.4% | 10.2% | 95.1% |
| 4 | 891 | 4.4h | 6.6% | 0 | 41.8% | 69.7% | 51.1% |
| 5 | 1395.5 | 10.9h | 0 | 4.8% | 51.1% | 27.9% | 69.7% |
| 6 | 785.2 | 4.2h | 0 | 0 | 11.3% | 50.1% | 87.6% |
| 7 | 856 | 7.6h | 23% | 0 | 41.9% | 29.5% | 50.1% |
| 8 | 899.5 | 7.1h | 0 | 34.8% | 51.1% | 27.9% | 69.7% |
| 9 | 761.8 | 5.9h | 12.1% | 8.1% | 17.2% | 46.6% | 66.3% |
| 10 | 1003 | 10.3h | 0 | 11.3% | 19.9% | 44.8% | 90.6% |

Because the problem scale is small in this test, some berths are not utilised in optimal solutions.

setting in Chapter 3, berth 5 here represents the berth with highest water level. Some vessels can only be moored at berths with a high water level. Therefore, the utilisation rates of berth 4 and 5 are usually high in the experimental results.

## 5.5 Conclusion

This chapter for the first time proposes a new discrete-event simulation framework which can be used to simulate and optimise different variants of optimisation problems. The contributions can be summarised as follows:

1. A case study of BPPs was conducted.
    - An online bin packing algorithm was developed and integrated to the simulation model.
    - A set of test cases were generated using the framework to evaluate the online algorithm.
    - The results of the online algorithm on these test cases were then compared with those of a static algorithm from the literature.
2. A case study of BAP was conducted.
    - A BAP model was built in the simulation and the input data were read externally.
    - The meta-heuristic LF-BAP proposed in Chapter 3 was integrated to the simulation model. With the results, the complex simulation model was able to assign vessels to the corresponding berths in order.
3. The results were visualised in 3D graphics and statistics in both case studies.
4. This framework is developed based on an open-source simulation software named JaamSim. In order to build the simulation models, various new objects were developed.

# Chapter 6

# Conclusion and future work

This chapter discusses whether the identified gaps have been at least partially filled by revisiting the questions raised in Section 1.4. The research started from answering questions about if there are any missing links between academic logistics optimisation research in port operations and real-world applications. After identifying some important but not-well studied issues, the rest of the thesis has focused on closing these gaps. As results of this research, some problems have been found rarely considered in academic optimisation. Solutions have been provided by using optimisation methods and simulation approach.

## 6.1 Summary of contributions

This research aims to develop realistic solutions to enhance the efficiency of port operations. Major contributions have been summarised below.

1. Identify the gaps between academic and real-world scenarios in BAPs and optimisation-simulation for ports, including the important characteristics that have not been covered in academic research, the problems that have not been well-investigated, and the information that we can use to solve and evaluate logistics optimisation problems in reality.

2. Develop novel approaches to solve discrete dynamic BAPs. The proposed approach takes into account practically important constraints which have not been considered in evolutionary computation. In order to solve the problem effectively, the new approaches are proposed based on the study of the problem characteristics, the strength and weaknesses of existing techniques, the performance assessment of each algorithmic component.

3. Define and develop a new framework. For the first time discrete-event simulation is used as a general platform to compare and visualise logistic optimisation problems at ports that closely reflect various characteristics of real-world situations. The framework also provides a useful tool for researchers in the field to integrate their optimisation algorithms in the developed framework to evaluate/compare their robust/dynamic optimisation algorithms.

## 6.2 Future work

With our proposed approaches many new research avenues relating to optimisation and simulation open up. The literature review in Section 2 also shows many open research areas. Some possible future research directions to better link academic optimisation problems to real-world applications are summarised.

*Solve hybrid BAPs with current meta-heuristics:* In Chapter 3, the reason for solving the BAP as a discrete problem has been clarified. A future work that can be carried out is to consider the length of the quay in BAPs to make the problem suitable for more real-world scenarios. To the best of our knowledge, there is no existing work dealing with hybrid BAPs with tidal constraints. With current meta-heuristics, the information of berth availability needs to be updated and the conflicts of berth occupancy need to be taken into account in the decoding process. In Chapters 3 and 4, the proposed meta-heuristics have shown promising results. In order to solve the extended problem efficiently, other strategies like the restart scheme can be potentially applied.

*Consider time windows and investigate other meta-heuristics:* In practical applications, time windows may be subjected to ports and vessels. In some situations, a port does not provide services all the times because it normally serves a number of shipping companies and there are cultural and social factors. Due to contractual agreements between shipping companies and the port, a vessel usually has to depart within a certain period after arrival. In order to achieve a feasible schedule, the time window is an important condition we need to take into account in the future. Since more constraints are taking into the problem, we also need to investigate other meta-heuristics. A variety of approximate methods have been proved the efficiency in the literature. Studying other meta-heuristics is meaningful for BAPs with multiple constraints.

*Robust BAPs with uncertainty:* According to the case study from our industrial partner, the arrival time of each vessel is estimated. However, such information can be affected by many factors. The same issue is also applied to handling time, etc. It has been taken into account in Golias et al. [2014], Golias [2011], Han et al. [2010], Sheikholeslami and Ilati [2017] in terms of uncertain arrival time and handling time. Furthermore, BAPs with QC assignment was also mentioned frequently in academic research, such as Giallombardo et al. [2010], Han et al. [2010], Zeng et al. [2011]. It sometimes affects the loading and unloading time of each vessel as well. Therefore, dealing with uncertainty and resource assignment can be meaningful as an extension of BAPs with tidal constraints.

*Simulation with optimisation for ports:* Various simulation models with integrated optimisation were studied in Chapter 2. A lack of flexible simulation framework is identified in Chapter 2. In Chapter 5, a user-friendly framework was proposed. Two models were built as demonstration to integrate optimisation approaches and simulate BAPs and BPPs. However, in order to develop the framework as a powerful tool for simulation and optimisation, further improvements are essential. Firstly, it has to be capable of comparing different optimisation algorithms and making decisions. Secondly, it should be able to assess the robustness of optimisation algorithms by modelling the uncertainty in the simulation. Moreover, it should allow combining BAPs with other logistic problems at ports such as vehicle routing and stowage planning.

# Bibliography

S. Abdullah and M. Abdolrazzagh-Nezhad. Fuzzy job-shop scheduling problems: A review. *Information Sciences*, 278:380–407, 2014.

Akanksha Gupta. The world's 10 biggest ports, 2013. URL `http://www.ship-technology.com/features/feature-the-worlds-10-biggest-ports/`.

A. F. Ali. A hybrid gravitational search with levy flight for global numerical optimization. *Information Sciences Letters Inf. Sci. Lett*, 4:71–83, 2015.

R. Alvarez-Valdes. Lower bounds for three-dimensional multiple-bin-size bin packing problems. *OR Spectrum*, 2013.

R. Alvarez-Valdés, F. Parreño, and J. M. Tamarit. A grasp/path relinking algorithm for two-and three-dimensional multiple bin-size bin packing problems. *Computers & Operations Research*, 40(12):3081–3090, 2013.

Andrew Mwaniki. Busiest Cargo Ports In South America, 2018. URL `https://www.worldatlas.com/articles/busiest-cargo-ports-in-south-america.html`.

P. Angeloudis and M. G. Bell. A review of container terminal simulation models. *Maritime Policy & Management*, 38(5):523–540, 2011.

C. Arango, P. Cortés, J. Muñuzuri, and L. Onieva. Berth allocation planning in seville inland port by simulation and optimisation. *Advanced Engineering Informatics*, 25 (3):452–461, 2011.

M. M. Baldi. The three-dimensional knapsack problem with balancing constraints. *Applied Mathematics and Computation*, 2012a.

M. M. Baldi. The generalized bin packing problem. *Transportation Research Part E*, 2012b.

V. H. Barros, T. S. Costa, A. C. Oliveira, and L. A. Lorena. Model and heuristic for berth allocation in tidal bulk ports with stock level constraints. *Computers & Industrial Engineering*, 60(4):606–613, 2011.

J. A. Bennell, L. S. Lee, and C. N. Potts. A genetic algorithm for two-dimensional bin packing with due dates. *International Journal of Production Economics*, 145(2): 547–560, 2013.

H. Beyer. Robust optimization - a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering, Elsevier*, 2007.

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

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

A. Borshchev and A. Filippov. From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In *Proceedings of the 22nd international conference of the system dynamics society*, volume 22. Citeseer, 2004.

W. Bortfeldt. Constraints in container loading - a state-of-the-art review. *European Journal of Operational Research*, 2012.

I. Boussaid. A survey on optimization metaheuristics. *Information Sciences, elsevier*, 2013.

C. Briano, E. Briano, A. G. Bruzzone, and R. Revetria. Models for support maritime logistics: a case study for improving terminal planning. In *Proceedings of the 19th European Conference on Modeling and Simulation (ECMS)*, pages 199–203, 2005.

K. Buhrkal, S. Zuglian, S. Ropke, J. Larsen, and R. Lusby. Models for the discrete berth allocation problem: A computational comparison. *Transportation Research Part E: Logistics and Transportation Review*, 47(4):461–473, 2011.

M. Burcea, P. W. H. Wong, and F. C. C. Yung. Online multi-dimensional dynamic bin packing of unit-fraction items. In *Algorithms and Complexity*, pages 85–96. Springer, 2013.

H. J. Carlo, I. F. Vis, and K. J. Roodbergen. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research*, 236(1):1–13, 2014.

S. Ceschia and A. Schaerf. Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, 19(2):275–294, 2013.

D. Chang, Z. Jiang, W. Yan, and J. He. Integrating berth allocation and quay crane assignments. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):975–990, 2010.

C. H. Che. The multiple container loading cost minimization problem. *European Journal of Operational Research*, 2011.

C. Y. Cheong, C. Lim, K. C. Tan, and D. Liu. A multi-objective evolutionary algorithm for berth allocation in a container port. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 927–934. IEEE, 2007.

C. Y. Cheong, K. C. Tan, D. Liu, and C. Lin. Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research*, 180(1):63–103, 2010.

China ports. Total number of ships entering and leaving port in China, 2014. URL `http://www.chinaports.com/portlspnews/003DC76DC7B54CD19C8C673BA2B80D6C/view`.

C. G. Christensen and C. T. Holst. Berth allocation in container terminals. *Master's thesis, Technical University of Denmark*, 2008.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. *Handbooks in operations research and management science*, 14:189–284, 2007.

R. Cimpeanu, M. T. Devine, D. Tocher, and L. Clune. Development and analysis of a port terminal loader model at rusal aughinish. *Simulation Modelling Practice and Theory*, 51:14 – 30, 2015. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2014.11.001. URL `http://www.sciencedirect.com/science/article/pii/S1569190X14001634`.

U. Clausen and J. Kaffka. Development of priority rules for handlings in inland port container terminals with simulation. *Journal of Simulation*, 10(2):95–102, 2016.

J.-F. Cordeau, G. Laporte, P. Legato, and L. Moccia. Models and tabu search heuristics for the berth-allocation problem. *Transportation science*, 39(4):526–538, 2005.

J.-F. Cordeau, P. Legato, R. M. Mazza, and R. Trunfio. Simulation-based optimization for housekeeping in a container transshipment terminal. *Computers & Operations Research*, 53:81–95, 2015.

M. d. G. Costa and M. E. Captivo. Weight distribution in container loading: a case study. *International Transactions in Operational Research*, 23(1-2):239–263, 2016.

Crainic. Efficient lower bounds and heuristics for the variable cost and size bin. *Computers & Operations Research*, 2011.

T. G. Crainic. Bin packing problems with uncertainty on item characteristics: an application to capacity planning in logistics. *Social and Behavioral Sciences*, 2014.

T. G. Crainic, G. Perboli, and R. Tadei. Ts 2 pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760, 2009.

A. Dadashi, M. A. Dulebenets, M. M. Golias, and A. Sheikholeslami. A novel continuous berth scheduling model at multiple marine container terminals with tidal considerations. *Maritime Business Review*, 2(2):142–157, 2017.

G. Dagkakis and C. Heavey. A review of open source discrete event simulation software for operations research. *Journal of Simulation*, 2015.

T. Davidovic, N. Kovac, and Z. Stanimirovic. Vns-based approach to minimum cost hybrid berth allocation problem. In *Proc. XLII International Symposium on Operations Research, SYMOPIS*, pages 237–240, 2015.

A. de Almeida. A particular approach for the three-dimensional packing problem with additional constraints. *Computer and Opertion Research*, 2010.

P. de Langen, M. Nidjam, and M. van der Horst. New indicators to measure port performance. *Journal of Maritime Research*, 4(1):23–36, 2007.

A. D. De León, E. Lalla-Ruiz, B. Melián-Batista, and J. M. Moreno-Vega. A machine learning-based system for berth scheduling at bulk terminals. *Expert Systems with Applications*, 87:170–182, 2017.

R. M. de Oliveira, G. R. Mauri, and L. A. N. Lorena. Clustering search for the berth allocation problem. *Expert Systems with Applications*, 39(5):5499–5505, 2012.

B. Dragović, E. Tzannatos, and N. K. Park. Simulation modelling in ports and container terminals: literature overview and analysis by research field, application area and tool. *Flexible Services and Manufacturing Journal*, 29(1):4–34, 2017.

J. Egeblad and D. Pisinger. Heuristic approaches for the two-and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36(4):1026–1049, 2009.

M. Eley. A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 2003.

L. Epstein and M. Levy. Dynamic multi-dimensional bin packing. *Journal of Discrete Algorithms*, 8(4):356–372, 2010.

L. Epstein, L. M. Favrholdt, and J. S. Kohrt. Comparing online algorithms for bin packing problems. *Journal of Scheduling*, 15(1):13–21, 2012.

X. Feng. Hybrid genetic algorithms for the three-dimensional multiple container packing problem. *Flexible Services and Manufacturing - Springer*, 2013.

ForConstructionPros. Turning Sea Into Land, 2016. URL `https://www.forconstructionpros.com/concrete/equipment-products/article/12278666/turning-sea-into-land`.

G. Giallombardo, L. Moccia, M. Salani, and I. Vacca. Modeling and solving the tactical berth allocation problem. *Transportation Research Part B: Methodological*, 44(2): 232–245, 2010.

J. Göbel, P. Joschko, A. Koors, and B. Page. The discrete event simulation framework desmo-j: Review, comparison to other frameworks and latest development. In *ECMS*, 2013.

M. Golias, M. Boile, and S. Theofanis. The berth allocation problem: A formulation reflecting service delay penalties and early premiums. Technical report, 2007.

M. Golias, I. Portal, D. Konur, E. Kaisar, and G. Kolomvos. Robust berth scheduling at marine container terminals via hierarchical optimization. *Computers & Operations Research*, 41:412–422, 2014.

M. M. Golias. A bi-objective berth allocation formulation to account for vessel handling time uncertainty. *Maritime Economics & Logistics*, 13(4):419–441, 2011.

M. M. Golias and H. E. Haralambides. Berth scheduling with variable cost functions. *Maritime Economics & Logistics*, 13(2):174–189, 2011.

M. M. Golias, M. Boile, and S. Theofanis. Berth scheduling by customer service differentiation: A multi-objective approach. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):878–892, 2009a.

M. M. Golias, G. K. Saharidis, M. Boile, S. Theofanis, and M. G. Ierapetritou. The berth allocation problem: Optimizing vessel arrival time. *Maritime Economics & Logistics*, 11(4):358–377, Dec 2009b. ISSN 1479-294X. doi: 10.1057/mel.2009.12. URL `https://doi.org/10.1057/mel.2009.12`.

M. M. Golias, M. Boile, and S. Theofanis. A lamda-optimal based heuristic for the berth scheduling problem. *Transportation Research Part C: Emerging Technologies*, 18(5): 794–806, 2010.

Y. Guan and R. K. Cheung. The berth allocation problem: models and solution methods. *OR spectrum*, 26(1):75–92, 2004.

Y. Guan, W.-Q. Xiao, R. K. Cheung, and C.-L. Li. A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis. *Operations Research Letters*, 30(5):343–350, 2002.

M. Gutowski. Levy flights as an underlying mechanism for global optimization algorithms. *ArXiv Mathematical Physics e-prints*, June 2001.

X. Han, Z. Lu, and L. Xi. A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. *European Journal of Operational Research*, 207(3):1327 – 1340, 2010. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2010.07.018. URL `http://www.sciencedirect.com/science/article/pii/S037722171000528X`.

P. Hansen and C. Oguz. *A note on formulations of the static and dynamic berth allocation problems.* Citeseer, 2003.

P. Hansen, C. Oğuz, and N. Mladenović. Variable neighborhood search for minimum cost berth allocation. *European Journal of Operational Research*, 191(3):636–649, 2008.

J. He, W. Zhang, Y. Huang, and W. Yan. A simulation optimization method for internal trucks sharing assignment among multiple container terminals. *Advanced Engineering Informatics*, 27(4):598–614, 2013.

J. He, Y. Huang, and D. Chang. Simulation-based heuristic method for container supply chain network optimization. *Advanced Engineering Informatics*, 29(3):339–354, 2015a.

J. He, Y. Huang, and W. Yan. Yard crane scheduling in a container terminal for the trade-off between efficiency and energy consumption. *Advanced Engineering Informatics*, 29(1):59–75, 2015b.

J. He, Y. Huang, W. Yan, and S. Wang. Integrated internal truck, yard crane and quay crane scheduling in a container terminal considering energy consumption. *Expert Systems with Applications*, 42(5):2464–2487, 2015c.

V. Hemmelmayr. Variable neighbourhood search for the variable sized bin packing problem. *Computers & Operations Research*, 2012.

M. Hendriks, M. Laumanns, E. Lefeber, and J. T. Udding. Robust periodic berth planning of container vessels. In *Proc. of the Third German Korean Workshop on Container Terminal Management: IT-based Planning and Control of Seaport Container Terminals and Transportation Systems*, pages 1–13, 2008.

Z.-H. Hu. Multi-objective genetic algorithm for berth allocation problem considering daytime preference. *Computers & Industrial Engineering*, 89:2–14, 2015.

G. Ilati, A. Sheikholeslami, and E. Hassannayebi. A simulation-based optimization approach for integrated port resource allocation problem. *PROMET-Traffic&Transportation*, 26(3):243–255, 2014.

T. V. L. N. Ilkyeong Moon. Container packing problem with balance constraints. *OR Spectrum*, 2013.

A. Imai, K. Nagaiwa, and C. W. Tat. Efficient planning of berth allocation for container terminals in asia. *Journal of advanced transportation*, 31(1):75–94, 1997.

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

A. Imai, E. Nishimura, and S. Papadimitriou. Berth allocation with service priority. *Transportation Research Part B: Methodological*, 37(5):437–457, 2003.

A. Imai, X. Sun, E. Nishimura, and S. Papadimitriou. Berth allocation in a container port: using a continuous location space approach. *Transportation Research Part B: Methodological*, 39(3):199–221, 2005.

A. Imai, E. Nishimura, M. Hattori, and S. Papadimitriou. Berth allocation at indented berths for mega-containerships. *European Journal of Operational Research*, 179(2): 579–593, 2007.

A. Imai, E. Nishimura, and S. Papadimitriou. Marine container terminal configurations for efficient handling of mega-containerships. *Transportation Research Part E: Logistics and Transportation Review*, 49(1):141–158, 2013.

A. Imai, Y. Yamakawa, and K. Huang. The strategic berth template problem. *Transportation Research Part E: Logistics and Transportation Review*, 72:77–100, 2014.

K. Jansen and K.-M. Klein. A robust afptas for online bin packing with polynomial migration. In F. V. Fomin, R. Freivalds, M. Kwiatkowska, and D. Peleg, editors, *Automata, Languages, and Programming*, pages 589–600, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39206-1.

B. Ji, X. Yuan, and Y. Yuan. Modified nsga-ii for solving continuous berth allocation problem: Using multiobjective constraint-handling strategy. *IEEE transactions on cybernetics*, 47(9):2885–2895, 2017.

J. Jiang and L. Cao. A hybrid simulated annealing algorithm for three-dimensional multi-bin packing problems. In *Systems and Informatics (ICSAI), 2012 International Conference on*, pages 1078–1082. IEEE, 2012.

Y. Jin and B. Sendhoff. Constructing dynamic optimization test problems using the multi-objective optimization concept. In *Applications of Evolutionary Computing*, pages 525–536. Springer, 2004.

L. Junqueira. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 2012.

L. Junqueira. Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier. *Computers & Industrial Engineering*, 2015.

S. Kavakeb, T. T. Nguyen, Z. Yang, and I. Jenkinson. Evolutionary fleet sizing in static and uncertain environments with shuttle transportation tasks - the case studies of container terminals. *IEEE Computational Intelligence Magazine, in press*, 2016.

K. H. Kim and K. C. Moon. Berth scheduling by simulated annealing. *Transportation Research Part B: Methodological*, 37(6):541–560, 2003.

D. King. The fastest simulation software, 2014. URL `https://jaamsim.blogspot.com/2014/11/the-fastest-simulation-software.html`.

D. King and H. S. Harrison. Jaamsim open-source simulation software. In *Proceedings of the 2013 Grand Challenges on Modeling and Simulation Conference*, page 1. Society for Modeling & Simulation International, 2013.

C. Kontovas and H. N. Psaraftis. Reduction of emissions along the maritime intermodal container chain: operational models and policies. *Maritime Policy & Management*, 38(4):451–469, 2011.

N. Kovač. Metaheuristic approaches for the berth allocation problem. *Yugoslav Journal of Operations Research*, 27(3):265–289, 2017.

E. Lalla-Ruiz and S. Voß. Popmusic as a matheuristic for the berth allocation problem. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):173–189, 2016.

E. Lalla-Ruiz, B. Melián-Batista, and 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):1132–1141, 2012.

E. Lalla-Ruiz, J. L. González-Velarde, B. Melián-Batista, and J. M. Moreno-Vega. Biased random key genetic algorithm for the tactical berth allocation problem. *Applied Soft Computing*, 22:60–76, 2014.

E. Lalla-Ruiz, S. Voß, C. Expósito-Izquierdo, B. Melián-Batista, and J. M. Moreno-Vega. A popmusic-based approach for the berth allocation problem under time-dependent limitations. *Annals of Operations Research*, pages 1–27, 2015.

E. Lalla-Ruiz, C. Expósito-Izquierdo, B. Melián-Batista, and 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):1001–1012, 2016.

D.-H. Lee and J. G. Jin. Feeder vessel management at container transshipment terminals. *Transportation Research Part E: Logistics and Transportation Review*, 49(1):201–216, 2013.

D.-H. Lee and H. Q. Wang. Integrated discrete berth allocation and quay crane scheduling in port container terminals. *Engineering Optimization*, 42(8):747–761, 2010.

D.-H. Lee, J. H. Chen, and J. X. Cao. The continuous berth allocation problem: A greedy randomized adaptive search solution. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):1017–1029, 2010.

D.-H. Lee, J. G. Jin, and J. H. Chen. Terminal and yard allocation problem for a container transshipment hub with multiple terminals. *Transportation Research Part E: Logistics and Transportation Review*, 48(2):516–528, 2012.

P. Legato, D. Gullì, and R. Trunfio. The quay crane deployment problem at a maritime container terminal. In *Submitted to the 22th European Conference on Modelling and Simulation*, 2008.

P. Legato, R. M. Mazza, and R. Trunfio. Simulation-based optimization for discharge/loading operations at a maritime container terminal. *OR spectrum*, 32(3): 543–567, 2010.

P. Legato, R. M. Mazza, and D. Gullì. Integrating tactical and operational berth allocation decisions via simulation–optimization. *Computers & Industrial Engineering*, 78: 84–94, 2014.

H. Li and D. Wang. Parallel simulation-based optimization on block planning and dynamic truck configuration of container terminals. *International Journal of Information*, 4(2):1–8, 2009.

Y. Li. A compromised large-scale neighborhood search heuristic for capacitated air cargo loading planning. *European Journal of Operational Research*, 2009.

A. Lim. The berth planning problem. *Operations research letters*, 22(2):105–110, 1998.

A. Lim. The single container loading problem with axle weight constraints. *International Journal of Production Economics*, 2013.

A. Lim and X. Zhang. The container loading problem. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 913–917. ACM, 2005.

S.-W. Lin and C.-J. Ting. Solving the dynamic berth allocation problem by simulated annealing. *Engineering Optimization*, 46(3):308–327, 2014.

J. Liu. A novel hybrid tabu search approach to container loading. *Computers & Operations Research*, 2011.

W. G. Liying Songa, Tom Cherrettb. Study on berth planning problem in a container seaport: Using an integrated programming approach. *Computer and Industrial Engineering*, 2012.

A. Lodi, S. Martello, and D. Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4): 345–357, 1999.

H. Ma, S. Chung, H. Chan, and L. Cui. An integrated model for berth and yard planning in container terminals with multi-continuous berth layout. *Annals of Operations Research*, pages 1–23, 2017.

MarineLink. Collision closes houston ship channel, 2015. URL `https://www.marinelink.com/news/collision-houston-channel387314`.

S. Martello. Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software*, 2007.

A. Martinez-Sykora, R. Alvarez-Valdes, J. Bennell, and J. M. Tamarit. Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts. *Omega*, 52:15–32, 2015.

G. R. Mauri, G. M. Ribeiro, L. A. N. Lorena, and G. Laporte. An adaptive large neighborhood search for the discrete and continuous berth allocation problem. *Computers & Operations Research*, 70:140–154, 2016.

P. Meersmans and R. Dekker. Operations research supports container handling (econometric institute report 234). *Erasmus University Rotterdam*, 2001.

Z. Meng, H. Shen, and T. Zhao. Hybrid artificial bee colony algorithm based on cuckoo search strategy. In *Semantics, Knowledge and Grids (SKG), 2016 12th International Conference on*, pages 136–140. IEEE, 2016.

Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

M. F. Monaco and M. Sammarra. The berth allocation problem: a strong formulation solved by a lagrangean approach. *Transportation Science*, 41(2):265–280, 2007.

R. Moorthy and C.-P. Teo. Berth management in container terminal: the template design problem. In *Container Terminals and Cargo Systems*, pages 63–86. Springer, 2007.

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):333–380, 1937. ISSN 00804614. URL `http://www.jstor.org/stable/91337`.

T. T. Nguyen. *Continuous dynamic optimisation using evolutionary algorithms*. PhD thesis, University of Birmingham, 2011.

T. T. Nguyen and X. Yao. Benchmarking and solving dynamic constrained problems. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 690–697. IEEE, 2009a.

T. T. Nguyen and X. Yao. Dynamic time-linkage problems revisited. In *Applications of Evolutionary Computing*, pages 735–744. Springer, 2009b.

T. T. Nguyen and X. Yao. Continuous dynamic constrained optimization - the challenges. *IEEE Transactions on Evolutionary Computation*, 16(6):769–786, 2012.

T. T. Nguyen and X. Yao. Dynamic time-linkage evolutionary optimization: Definitions and potential solutions. In *Metaheuristics for Dynamic Optimization*, pages 371–395. Springer, 2013.

T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

T. Nishi, T. Okura, E. Lalla-Ruiz, and S. Voß. A dynamic programming-based matheuristic for the dynamic berth allocation problem. *Annals of Operations Research*, pages 1–20, 2017.

E. Nishimura, A. Imai, and S. Papadimitriou. Berth allocation planning in the public berth system by genetic algorithms. *European Journal of Operational Research*, 131 (2):282–292, 2001.

NWEUROPE. Opportunities for Territorial Change, 2008. URL `http://www.espace-project.org/publications/IIIBPublicationonBestIIIBprojects.pdf`.

A. Ouaarab, B. Ahiod, and X.-S. Yang. Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications*, 24(7-8):1659–1669, 2014.

C. Ozguven, L. Ozbakir, and Y. Yavuz. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, 34(6):1539–1548, 2010.

C. Paquay. Three dimensional bin packing problem applied to air cargo. *Colloque SIL 2011*, 2011.

C. Paquay. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *Intl. Trans. in Op. Res*, 2014.

K. Park and K. H. Kim. Berth scheduling for container terminals by using a sub-gradient optimization technique. *Journal of the operational research society*, 53(9):1054–1062, 2002.

Y.-M. Park and K. H. Kim. A scheduling method for berth and quay cranes. In *Container Terminals and Automated Transport Systems*, pages 159–181. Springer, 2005.

Peel Ports Group. Mersey Ports Master Plan, 2016. URL `https://www.peelports.com/about/master-plan`.

J. Peng and B. Zhang. Bin packing problem with uncertain volumes and capacities, 2012.

G. Perboli, R. Tadei, and M. M. Baldi. The stochastic generalized bin packing problem. *Discrete Applied Mathematics*, 160(7):1291–1297, 2012.

D. Pisinger and M. Sigurd. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2):154–167, 2005.

Port of Antwerp. The Port of Antwerp The supermarket of Europe , 2014. URL `http://www.portofantwerp.com/sites/portofantwerp/files/POA-1370_Publiekskaart_EN-06102014_0.pdf`.

T. Qin, Y. Du, and 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*, 87:167–185, 2016.

D. P. K. Reuven Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc, 2008.

H. Richter. Memory design for constrained dynamic optimization problems. In *Applications of Evolutionary Computation*, pages 552–561. Springer, 2010.

M. Rodriguez-Molins, L. Ingolotti, F. Barber, M. A. Salido, M. R. Sierra, and J. Puente. A genetic algorithm for robust berth allocation and quay crane assignment. *Progress in Artificial Intelligence*, 2(4):177–192, 2014.

G. Saharidis, M. Golias, M. Boile, S. Theofanis, and M. Ierapetritou. The berth scheduling problem with customer differentiation: a new methodological approach based on hierarchical optimization. *The International Journal of Advanced Manufacturing Technology*, 46(1-4):377–393, 2010.

M. A. Salido, M. Rodriguez-Molins, and F. Barber. Integrated intelligent techniques for remarshaling and berthing in maritime terminals. *Advanced Engineering Informatics*, 25(3):435–451, 2011.

M. A. Salido, M. Rodriguez-Molins, and F. Barber. A decision support system for managing combinatorial problems in container terminals. *Knowledge-Based Systems*, 29:63–74, 2012.

A. Sheikholeslami and R. Ilati. A sample average approximation approach to the berth allocation problem with uncertain tides. *Engineering Optimization*, pages 1–17, 2017.

A. Sheikholeslami, G. Ilati, and M. Kobari. The continuous dynamic berth allocation problem at a marine container terminal with tidal constraints in the access channel. *International Journal of Civil Engineering*, 12(3):344–353, 2014.

X.-N. Shen and X. Yao. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences*, 298:198–224, 2015.

S. Sivanandam and S. Deepa. *Introduction to genetic algorithms*. Springer Science & Business Media, 2007.

R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR spectrum*, 30(1):1–52, 2008.

D. Steenken, S. Voß, and R. Stahlbock. Container terminal operation and operations research-a classification and literature review. *OR spectrum*, 26(1):3–49, 2004.

D. K. Sutantyo, S. Kernbach, P. Levi, and 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*, pages 1–6. IEEE, 2010.

J. Swain. Simulation software survey. *INFORMS OR/MS Today*, pages 42–51, 2013.

Ta Kung Pao. Capesize Freight Route High, 2011. URL `http://www.hh-ship.com/hh/en/newsshow.asp?id=35`.

É. D. Taillard and S. Voß. Popmusic partial optimization metaheuristic under special intensification conditions. In *Essays and surveys in metaheuristics*, pages 613–629. Springer, 2002.

G. Terdik and T. Gyires. Lévy flights and fractal modeling of internet traffic. *IEEE/ACM Transactions on Networking*, 17(1):120–129, 2009.

S. Theofanis, M. Boile, and M. Golias. An optimization based genetic algorithm heuristic for the berth allocation problem. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4439–4445. IEEE, 2007.

T. Tian, W. Zhu, A. Lim, and L. Wei. The multiple container loading problem with preference. *European Journal of Operational Research*, 2015.

timeanddate.com. The Moon's Effect on Ocean Tides, 2018. URL `https://www.timeanddate.com/astronomy/moon/tides.html`.

C.-J. Ting, K.-C. Wu, and H. Chou. Particle swarm optimization algorithm for the berth allocation problem. *Expert Systems with Applications*, 41(4):1543–1550, 2014.

T. Tran, T. T. Nguyen, and H. L. Nguyen. Global optimization using lévy flights. In *Proceedings of ICT.rda'04*, Hanoi, Sept. 2004.

N. Umang, M. Bierlaire, and I. Vacca. The berth allocation problem in bulk ports. In *11th Swiss Transport Research Conference*, number EPFL-CONF-167446, 2011.

N. Umang, M. Bierlaire, and I. Vacca. Exact and heuristic methods to solve the berth allocation problem in bulk ports. *Transportation Research Part E: Logistics and Transportation Review*, 54:14–31, 2013.

I. Vacca, M. Bierlaire, and M. Salani. Optimization at container terminals: status, trends and perspectives. In *Swiss Transport Research Conference*, number TRANSP-OR-CONF-2006-013, 2007.

Vesseltracker. Vesseltracker.com, 2018. URL `http://www.vesseltracker.com/en/Home.html`.

I. F. Vis and R. De Koster. Transshipment of containers at a container terminal: An overview. *European journal of operational research*, 147(1):1–16, 2003.

G. Viswanathan, E. Raposo, and M. Da Luz. Lévy flights and superdiffusion in the context of biological encounters and random searches. *Physics of Life Reviews*, 5(3): 133–150, 2008.

F. Wang and A. Lim. A stochastic beam search for the berth allocation problem. *Decision Support Systems*, 42(4):2186–2196, 2007.

G. Wascher, H. Haussner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

K. Weicker and N. Weicker. Dynamic rotation and partial visibility. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 1125–1131, 2000.

Wikipedia. List of busiest container ports, 2017. URL `https://en.wikipedia.org/wiki/List_of_busiest_container_ports`.

P. W. H. Wong and F. C. C. Yung. Competitive multi-dimensional dynamic bin packing via l-shape bin packing. In *Approximation and Online Algorithms*, pages 242–254. Springer, 2010.

World Shipping Council. TOP 50 WORLD CONTAINER PORTS, 2015. URL `http://www.worldshipping.org/about-the-industry/global-trade/top-50-world-container-ports`.

X. Xiang, C. Liu, and L. Miao. A bi-objective robust model for berth allocation scheduling under uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, 106:294–319, 2017.

D. Xu, C.-L. Li, and J. Y.-T. Leung. Berth allocation with time-dependent physical limitations on vessels. *European Journal of Operational Research*, 2012a.

Y. Xu, Q. Chen, and X. Quan. Robust berth scheduling with uncertain vessel delay and handling time. *Annals of Operations Research*, 192(1):123–140, 2012b.

J. B. Y Jin. Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE*, 2005.

C. Yang, X. Wang, and Z. Li. An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal. *Computers & Industrial Engineering*, 63(1):243–253, 2012.

X.-S. Yang and S. Deb. Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE, 2009.

T. Yu, Z. Qiang, and Z. Benfei. A genetic algorithm based on spatiotemporal conflict between continuous berth-allocation and time-varying specific crane assignment. *Engineering Optimization*, pages 1–22, 2018.

W. Yuping, X. Zhe, H. Youfang, H. Yangyang, and G. Tianyi. Study of continuous berth allocation algorithm based on fairness maximization. *Journal of Engineering Science & Technology Review*, 10(5), 2017.

E. Zehendner, G. Rodriguez-Verjan, N. Absi, S. Dauzère-Pérès, and D. Feillet. Optimized allocation of straddle carriers to reduce overall delays at multimodal container terminals. *Flexible Services and Manufacturing Journal*, 27(2-3):300–330, 2015.

Q. Zeng, Z. Yang, and X. Hu. Disruption recovery model for berth and quay crane scheduling in container terminals. *Engineering Optimization*, 43(9):967–983, 2011.

Q. Zeng, A. Diabat, and Q. Zhang. A simulation optimization approach for solving the dual-cycling problem in container terminals. *Maritime Policy & Management*, 42(8): 806–826, 2015.

Q. Zeng, Y. Feng, and Z. Chen. Optimizing berth allocation and storage space in direct transshipment operations at container terminals. *Maritime Economics & Logistics*, 19(3):474–503, 2017.

X. Zhao. A comparative review of 3d container loading algorithms. *International Transactions in Operational Research*, 2014.

L. Zhen. Tactical berth allocation under uncertainty. *European Journal of Operational Research*, 247(3):928–944, 2015.

L. Zhen, L. H. Lee, and E. P. Chew. A decision model for berth allocation under uncertainty. *European Journal of Operational Research*, 212(1):54–68, 2011.

P.-f. Zhou and H.-g. Kang. Study on berth and quay-crane allocation under stochastic environments in container terminal. *Systems Engineering-Theory & Practice*, 28(1): 161–169, 2008.

W. Zhu and A. Lim. A new iterative-doubling greedy–lookahead algorithm for the single container loading problem. *European Journal of Operational Research*, 222(3): 408–417, 2012.