Poznań University of Technology

Faculty of Computing and Telecommunications

Doctoral dissertation

Combinatorial optimization problems in port logistics

Jakub Wawrzyniak

Supervisor: prof. dr hab. eng. Maciej Drozdowski

Poznań, 2024

Acknowledgements

First and foremost, I am incredibly grateful to my supervisor, prof. dr hab. eng. Maciej Drozdowski, for his invaluable advice, continuous support, and patience during my PhD study. His immense knowledge and plentiful experience have encouraged me throughout my academic research and daily life.

Finally, I would like to express my gratitude to my wife, my children, and my parents. Without their tremendous understanding and encouragement in the past few years, it would have been impossible for me to complete my studies.

Contents

Chapter 1

Introduction

1.1 Motivation

Global container shipping plays a vital role in contemporary logistics landscape. It is the primary type of transport to move consumer goods between continents and countries. The expansion of various consumption behaviors, related to the opportunities offered by e-commerce, has significantly increased consumer expectations regarding the duration and possibilities of transporting goods, also between distant areas of the globe. Therefore, countries with the access to the coastline develop logistic industry and container vessel facilities in particular. This leads to a development of the global economy, but also growing complexity of transportation systems.

At the same time, we observe increasing competition, especially between large entities distributing various goods popular around the world. One example of such a situation is the so-called fast fashion. While leaving aside the ethical and ecological aspects, it is an ideal example that costs of producing new clothing have been reduced as much as possible, demand for product is well predictable, but the optimization of logistics costs remains an intensively explored field of competition. The above shows that logistics remains one of the key and the hardest sectors of the economy. In this context, maritime container transport is of great importance. There are well-known "world hubs" famous for their ports servicing this type of transport, e.g., Le Havre, Rotterdam, Singapore or Shanghai. In their daily work, these centers face typical problems

of the whole logistics industry, and additional difficulties are caused by problems specific to maritime transport and port support services. Such problems can be successfully mapped to issues widely researched and discussed in the areas of combinatorial optimization or operations research. It is therefore natural, that the experience gained from the research in these areas can and should be used to solve real business problems. The above leads to the motivation of this thesis, which is to conduct research in the areas of port logistics and combinatorial optimization.

1.2 Scope and Purpose

Ports and maritime container terminals are vital infrastructure to the global economy. Due to competition and increasing traffic, they are intensively optimized. There are many operations research formulations optimizing elements of port logistics: berth allocation problem, tugboat and quay crane assignment problems, landside container traffic routing [7, 8, 11, 14, 42, 27, 36, 41, 45, 59, 81, 88, 87].

When designing a new terminal or restructuring an existing one, a pivotal decision is partitioning the quay into berths. This is a subject of the quay partitioning problem (QPP) studied in this thesis. Although berths may seem to exist only as virtual lines on the quay they are widely used. For instance, Fig.1.1 shows vessel automatic identification system (AIS) [97] receiver positions at the quays in Le Havre and Singapore over 2021. Fig.1.1a,b show ship positions clustered for centroids in berth positions declared by port operators. Each color represents ships at a different berth. Fig.1.1c,d show histograms of ship positions along the quays with 20m resolution. It can be verified in Fig.1.1c and Fig.1.1d that histograms of ship positions along the quays are clustered with apparently less frequent positions (indents) showing the ends of the berths. Berths they play important managerial role because they determine ship positions at the quay and simplify crane, tugboat, hawser management. The layout of berths determines the design of the storage yard, planing and running the terminal internal transportation network. Thus, although the ships can be positioned anywhere along the quay stretch, the concept of discrete berth is widely used in maritime container terminal operations.

There is one more reason that quay partitioning is relevant. In the classic berth allocation problem (BAP) [7, 8], that consists in managing berths and vessels in time, berth layout is assumed as given. However, berth number and sizes, as the input to the BAP, can be optimized

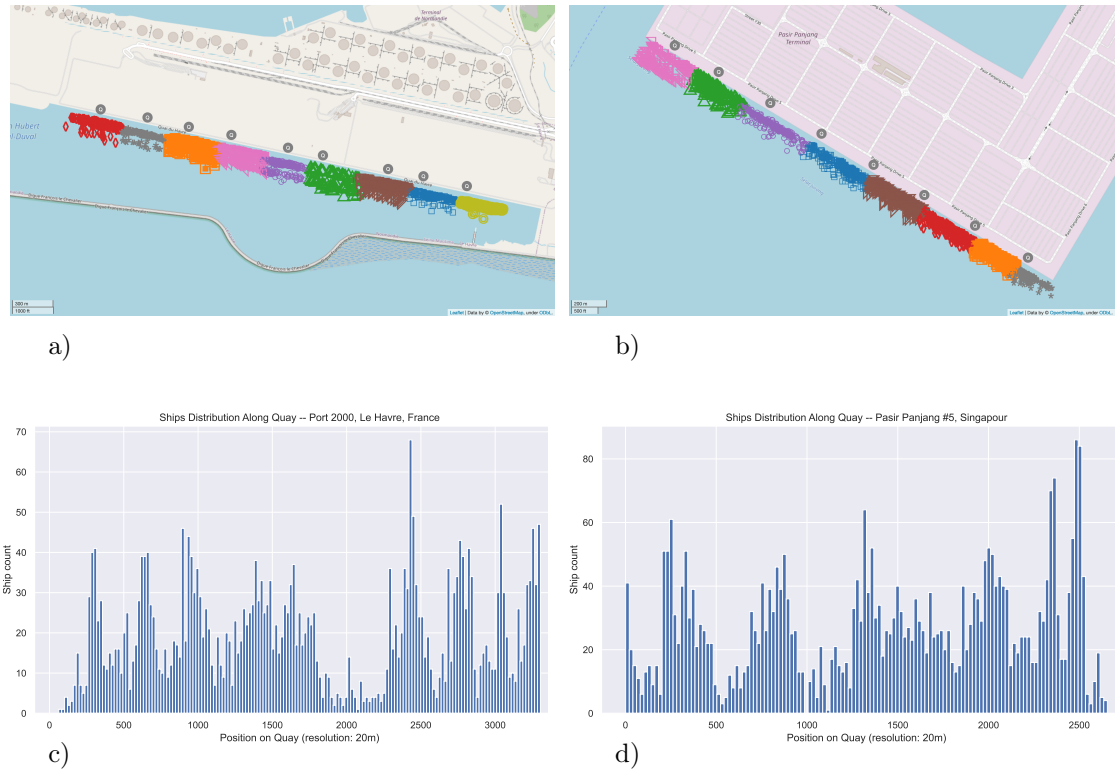


Figure 1.1: Examples of vessel clusters along a quay. a) map for Le Havre, b) map for Singapore, c) positions histogram Le Havre, d) positions histogram Singapore. In a) and b) "Q" denotes official berth position. Map data from OpenStreetMap openstreetmap.org/copyright.

for the terminal performance and do not have to be predetermined by the existing geographical and historical constraints of the terminal location. Quay partitioning is practically important when ports with long quays are designed. For example, Yangshan terminal has a single 5km-long quay and 30 berths [96]. Tuas Megaport in Singapore, when completed, is expected to have 26km of quays [99]. Thus, there is room for optimization of berth lengths. Partitioning a quay into berths should be considered an element of long-term strategic port planning, when a terminal is designed or an existing one is expanded. Since ports and container terminals are designed for many years to come, the consequences of bad quay partitioning can be costly. Hence, an obvious question emerges: what is the most efficient quay partitioning?

Although berths exist, partitioning a quay into berths has practical long-term implications and there is room for optimization, dividing a quay into berths has been studied very little. To

the best of our knowledge, there does not seem to exist any paper dealing with this problem, and there are reasons for that. Firstly, defining berths is more a logical than physical operation, and does not require extensive investments. It may seem a minor issue compared to the other costly matters of terminal design. Secondly, the designers and terminal management may implicitly choose the continuous vessel positioning model [7, 8], assuming that vessels are moored anywhere along the quay, and consequently, there are no disjoint berths. However, as we have already shown terminals throughout the world use berths (Fig.1.1). A further reason is that there are obvious solutions for quay partitioning, the quays are divided into equal berths of the largest expected vessel length. However, it is not certain that such obvious partitioning is the best one. Some alternative partitions might improve the overall performance of the terminal. So the question of berth layout is pertinent.

Technically, the berth layout of a terminal is a partitioning of its quays into berths. Thus, a solution of the QPP requires determining the number of quays and their lengths such that the stream of arriving container vessels waiting time is minimized. There should be no a priori condition on the number and lengths of the berths, except for the obvious ones: there is no need to have berths shorter than the shortest expected vessel, and there should be a limit on the maximum size of a berth (but this, as we shall see, depends on the type of berths considered). A key question this thesis wants to answer is the extent of the impact of quay partitioning on the overall performance of the container terminal. A further high-level goal of this thesis is to propose a set of algorithmic methods to better design quays in new ports with a special regard to maritime container terminals and their container vessel traffic. In this Ph.D. the following hypothesis is defended:

Maritime container terminal performance can be improved by optimization of quay partitioning into berths.

In order to achieve the above high-level goals, the work was divided into sub-problems focused on the components of Quay Partitioning Problem. The individual issues analyzed are:

I Deterministic Quay Partitioning Problem (DQPP) which assumes a single known future vessel arrival scenario. It will be assumed that either a single vessel can be moored in a berth, or at most two vessels can share a berth. The former assignment type will be called discrete or one-in-one layout (1in1 for short), the latter will be referred to as hybrid or two-in-one (2in1) layout. Two methods, based on mixed integer linear programming are

introduced to solve quay partitioning problem for the two aforementioned berth layouts. It will be demonstrated that 2in1 hybrid vessel mooring method offers a greater flexibility and reliability than the 1in1 discrete vessel mooring method. Therefore, 2in1 layout will be the basis for the analyses in the following study. Part of the results in this study were published in [91].

II Ship Traffic Model (STM) where a model of container ship arrival is proposed. An in-depth data analysis of the features and dependencies in the historical container vessel traffic data of eight world ports allowed to construct generators of test instances representative of the real traffic [90].

III Algorithm Selection Problem (ASP) which proposes a method of selecting a portfolio of algorithms solving Berth Allocation Problem (BAP). Evaluation of a particular solution of the QPP, i.e. of the quay partition, for a given vessel arrival scenario consists in assigning the arriving vessels to berths and assessing quality of the vessel service. This is equivalent to a classic maritime logistic Berth Allocation Problem. As will be shown in the following Section 2.3, scheduling methods considered in the current literature are not capable of solving instance sizes emerging in QPP. Consequently, a dedicated method selecting a portfolio of BAP algorithms which can be run in parallel, in limited time, offering best solutions in the given time limit, is proposed (see [88]). The elected algorithm portfolio is used in the next part of the study (SQPP).

IV Stochastic Quay Partitioning Problem (SQPP) This part of the thesis builds on the ship traffic model and and algorithm portfolios in the optimization of quay partitioning in the stochastic setting. The uncertainty of future vessel arrivals is represented by the STM. A set of algorithms is proposed for solving a stochastic version of the QPP. The solutions, that is the quay partitions, are evaluated by a portfolio of BAP algorithms solving a set of scenarios generated from the STM. Several types of computational experiments were conducted to discover the best solution features. The scalability and reliability of QPP solution workflow is analyzed.

1.3 Methodology

This chapter outlines the methodology used to analyze the above-mentioned optimization problems. Each problem presents unique challenges in the realm of maritime logistics and computational efficiency. The approaches are structured to address particular challenges through a combination of mathematical modeling, algorithmic development, and computational experiments.

Since problems considered in this dissertation have combinatorial nature, they are inherently difficult to solve. Therefore, computational complexity theory will be used as a guiding methodological framework [25]. Further steps will be added to handle ship arrival uncertainty represented by stochastic traffic model, and consequently, to include also stochastic optimization. This implies that the following steps will be undertaken:

1. Determining computational complexity of quay partitioning problem in the discrete version (DQPP). It will be shown that DQPP is **NP**-hard.

2. Analysis of real traffic data to prepare stochastic ship traffic model (STM) and generators of test instances representative for the real traffic.

3. Construction of exact algorithms. This approach will be used to analyze features of high-quality QPP solutions. Two types of exact algorithms are applicable:

   - Full Enumeration which explores all possible solutions. The primary advantage is certainty – these methods guarantee finding the optimal solution. Thus, full enumeration can provide benchmark solutions for comparisons with other methods. The major drawback is computational impracticality for large-scale problems due to the exponential time complexity. This relegates full enumeration to solving small scale problems.
   - Mixed Integer Linear Programming (MIP) which involve formulating the problem as a linear program with integer variables. MIP methods are powerful and versatile, able to provide exact solutions, have well-established theory and robust solvers. The primary disadvantage of MIP is its computational complexity for large-scale problems.

4. Construction of heuristic algorithms for SQPP. Though heuristics do not guarantee obtaining optimal solutions, their advantage is low computational complexity. Two types of

heuristics are of interest:

- Greedy algorithms which work by greedily making a locally optimal choice at each step of iterative solution construction. They are known for simplicity and speed. The downside is that can be myopic, overlooking non-greedy choices. In a typical setting a greedy algorithm constructs one solution and stops.
- Metaheuristics which are high-level construction schemes for algorithms exploring the considered problem solution space. Examples include Genetic Algorithms, Simulated Annealing, and Tabu Search. The primary advantage is their flexibility and applicability to a wide range of problems, including those with complex, non-linear, or multi-modal solution space landscapes. They are particularly useful for large-scale problems where exact methods are impractical. Metaheuristic complexity is well controlled because its runtime can be easily set by the algorithm designer. The downside is, e.g, setting them to work effectively and extensive parameter tuning. In a typical setting metaheuristics improve solutions over time and can provide many increasingly better solutions in the allowed runtime.

Heuristic methods are useful in two ways when solving SQPP:

- In the evaluation of quay partition for a scenario generated from STM, which means assigning arriving vessels to the berths for some minimum waiting time. This is equivalent to solving a classic Berth Allocation Problem (BAP) which is **NP**-hard itself and (unless **P**=**NP**) requires efficient heuristics.
- In constructing the quay partition, which is also combinatorial optimization problem in stochastic setting (**NP**)-hard for the discrete version).

5. Testing of DQPP and SQPP algorithms for runtime and solution quality. Due to the complexity of these problems, one of the key issues is the trade-off between the running time and the solution quality offered by a given algorithm. This should be taken into account, especially when time budget is limited.

6. Analysis of solution features both in the deterministic and in the stochastic setting.

1.4 Outline of the Thesis

Further organization of the thesis is the following. In Chapter 2 related literature is surveyed with particular focus on port design, berth allocation, ship traffic modeling and algorithm selection. The rest of the dissertation is divided into chapters devoted to the previously outlined sub-problems.

Chapter 3 is dedicated to the discrete quay partitioning problem (DQPP). In the Section 3.2 the problem is formulated. Sections 3.3 and 3.4 introduce mixed integer linear programs for 1in1 and 2in1 versions of DQPP. Section 3.5 discusses scalability of the proposed methods, e.g., compares 2in1 against 1in1 variants of QPP.

In Chapter 5 the problem of developing the ship traffic model (STM) for further simulation studies is discussed. The importance of developing such a model is justified in Section 5.1. Section 5.2 provides a concise view on the features of the STM and their relation with port logistics. In Section 5.3 the steps of building the STM are described. Section 5.4 introduces the algorithms constructing test instances adhering to the developed model. In Section 5.5 distinctive port features discovered in the course of constructing the STM, are analyzed.

Chapter 4 considers selecting algorithms for large berth allocation problems (BAPs). Variants of the BAP are discussed in Section 4.2. Sections 4.3, 4.4 and 4.5 contain, respectively, definitions of the algorithms for BAP, description of benchmarking datasets, and preliminary algorithm evaluation results. Section 4.6 introduces the algorithm selection problem and two portfolio selection methods: cover portfolios and regret portfolios. In Section 4.7 the two portfolio variants are analyzed on selected examples. Broader performance tests of the selected algorithm portfolios are summarized in Section 4.8.

In Chapter 6 stochastic quay partitioning problem (SQPP) is studied. In Section 6.1 this problem is formulated. Partition evaluation process is explained in Section 6.2. Section 6.3 introduces quay partitioning algorithms. Sections 6.4 and 6.5, respectively, describe QPP solution features and evaluation of partitioning algorithms. Technical aspects of the SQPP solution workflow operations are presented in Section 6.6.

The final Chapter 7 contains a summary, final conclusions and outlines options for further studies on quay partitioning. In the process of preparing this dissertation a number of additional technical reports and datasets were created, significantly exceeding acceptable volume of a Ph.D

thesis. Two technical reports are included as appendices to give the reader an insight into the results not included in the main text of the thesis.

Chapter 2

Related Work

This thesis builds on the research in port simulation, traffic modeling, berth allocation problem, and algorithm selection problem. Hence, this chapter provides a description of the state of the art in these areas, without claims to be exhaustive. The status of the research in the aforementioned subproblems is important to justify the approaches and to understand the design decisions made in this dissertation.

2.1 Maritime Container Terminal Design

Maritime ports are a major link of logistic chains. Hence, port performance determines both port attractiveness and the overall logistics efficiency. Port optimization has many dimensions, depending on the decision horizon (strategic, tactical or operational), position in space (quays, storage yards, inter-terminals areas, hinterland) and type of goods (containers, liquid, bulk). Not surprisingly, the related literature is huge. This thesis focuses on the strategic horizon, seaside area, and container handling. To the best of our knowledge, quay partitioning problem has not been posed before and the existing literature is related to QPP only in the very broad sense of maritime container terminal design. The focus of this study is on quay partitioning while BAP serves the purpose of evaluating a partition. Hence, this section is devoted to container terminal design. The literature mentioned in this section is collected and classified in Tab. 2.1.

Let us elaborate on time horizons. Classically in port optimization, strategic infrastructure

Table 2.1: Related work on maritime terminal design, research directions

[46]	review of container terminal design studies, regarding equipment choice, layout design for seaside, storage, hinterland areas, and evaluation methods, SL
	queuing models for berth numbers
[40]	survey of queuing theory models for berth number calculation
[30]	number of berths for Izmir port. SL, criterion: cost.
	seaside scheduling
[57]	determines **NP**-hardness of BAP
[7, 8]	BAP, QCAP literature reviews, TL,OL
[71]	review and taxonomy for BAP with uncertainty, TL,OL
[41]	berth schedule templates in time×space, TL, obj: cost and flow time
[10]	Laycan Allocation Problem – assigning berthing time windows to new vessels to charter, +BAP, +QCAP, TL,OL, obj: cost, alternatively, flow time
[63]	evaluation by simulation of four scenarios of sharing berths and quay cranes between by four terminal operators of a quay, TL,OL,multicriteria
[13]	evaluation by simulation of two alternative bulk and container terminal locations, SL, obj:(average waiting time)/(average service time)
	yard (storage) layout
[66]	evaluation by simulation of storage block width impact on quay crane rate and terminal performance, SL, obj: Gross Crane Rate (lifts/hour)
[95]	storage block width and length bi-criteria optimization by mathematical programming, SL, obj: cost, alternatively, makespan
[1]	integrated queuing network model for container unload process with ALVs, SL, obj: expected time to unload a container
[61]	a probabilistic model to predict storage space requirements, SL, obj: cost
[12]	deterministic performance analysis of single-/double-lane storage blocks, SL, bi-obj: container throughput and number of equipment units
[106]	two-stage stochastic programming model for storage blocks layout, with terminal operations simulations providing container transfer timing, SL, obj: investment and operational costs
[83]	PSO algorithm planning transition from diesel to green yard cranes, SL, obj: combined cost
	innovative designs and future technologies
[26]	review of future generation container terminal designs, SL
[43]	proposition of double-storey automated container port, SL
[47]	underground transportation system separating container movement from vessel operation, SL
[48]	survey of 32 offshore terminal and transshipment concepts and their combinations, including an idea of mobile harbor, SL

ALV – Automated Lifting Vehicles, BAP – Berth Allocation Problem, QCAP – Quay Crane Assignment Problem, QCSP – Quay Crane Scheduling Problem, SL, TL, OL – strategic, tactical and operational levels, respectively

decisions have an impact months or years after the decisions have been taken. Tactical decisions consider a time horizon of weeks and affect resource allocation, whereas operational decisions apply to the next few days or even hours. Terminal design at large belongs to the strategic level. We consider that building of berth templates formalized by [41] belongs to the tactical level (this can be assimilated to the laycan assignment problem, see [10]). A berth template is a makeshift pattern of vessel schedule (including the time interval and the berth to service them), built weeks or months before the actual vessel arrivals. Building such a template schedule may involve rejecting vessels if the schedule is too tight regarding terminal resources. The schedule is reevaluated after each contract between the terminal and the liner (and sometimes

the charterer, mainly for bulk cargo). Often ship liners propose cyclic arrivals [19], which simplifies the schedules and allows to build them months ahead. However, due to unavoidable uncertainties such schedules have to be rebuilt at the operational level, which leads to the classical BAP. Quay crane assignment and stowage planning are further seaside optimization problems, see [7, 8].

Let us return to terminal design, including terminal layout. [46] review and systematize 29 container terminal design studies regarding equipment choice, layout design for the seaside, storage, hinterland areas, and evaluation methods. They determine a typical sequence of design decisions from the equipment to the layout. [26] propose a study of the issues in container terminal design, retain 21 prominent papers and provide a valuable insight into future trends. A large majority of studies tackling particular elements of terminal design is based on simulation, with a few papers using queuing theory or optimization models. Simulation dominance can be attributed to the level of uncertainty and the complexity of the problem at hand (see [71]). Firstly, the future flow of vessels is inevitably uncertain. Still, the number of ships and their types, arrival frequencies, the number of containers they will load/unload, are necessary for dimensioning the terminal. Secondly, evaluating the efficiency of a given terminal layout, for a given traffic model, is already a difficult optimization problem [57]. Hence, finding the best layout is also computationally hard. Simulations rely on ship traffic models and evaluate the performance of several alternative designs (see for instance [66, 95]). Queuing models, e.g. for stacking containers [1], can cope with uncertainties, but are limited by the model complexity. Optimization models [61], [12] have difficulty taking uncertainty into account and still remaining tractable. Optimization models might remain limited to some specific sub-problems. It is also possible to couple optimization and simulation as in [106].

According to [26, 46], terminal design decomposes into berth layout for the seaside part, yard and traffic course layout for the yard part, gate and rail area layout for the landside part. Even though the landside issues should not be overlooked as fluid traffic to and from the hinterland is essential for the terminal competitiveness, while multi-modality is a way to limit greenhouse gas emissions, they are usually dealt with at the port authority level, not at the terminal level. Hence, a vast majority of the studies deal with yard layout. Many of them consider changes of the actual yards, optimizing the block sizes and orientation with respect to the wharfs and evaluating the effect of changing the container handling equipment. A number of papers propose

major changes in the yard structure: double-storey terminals like the Singapore project [43], underground transport for transshipment [47], container racks inspired by modern warehouses [48]. Planning the equipment of the yard is also an element of terminal design. A transit from diesel to electric, photovoltaic and hybrid yard cranes was studied by [83].

Concerning the seaside part, only a few papers consider berth layout. [63] proposed a simulation to verify if better allocation of the terminal resources significantly increase the performances. The size of the four berths is fixed, but the quay cranes may, or not, be shared among the berths according to the vessel lengths they serve. The results show that sharing quay cranes improves the performance, which is a property usually assumed in many later papers.

[13] proposed a simulation based on some ship traffic model considering different types of cargo, in a port reachable by a long waterway. Two scenarios considered two locations that can welcome either two berths for containers, or two berths for solid bulk. The choice criterion depends on the sum of shipping waiting times. Hence, despite the paper title which explicitly refers to berth layout, [13] essentially consider the impact of choosing a location for terminals, but not the berth layout of a given terminal with a fixed location.

In order to determine the number of berths for minimum ship waiting and idle berth costs, queuing theory was used. In [40] a survey of queuing theory models for berth number calculation is given. An M|M|m model is used to derive a relationship between the number of berths m and traffic intensity. Such a queuing theory approach is a simple model of a far more complex process because, e.g., specific ship size class relationship with the arrival and service times is neglected, scheduling disciplines are very basic, berths are identical, one ship occupies a whole berth. In [30] number of berths for Izmir port without distinguishing specific type of the container, cargo and passenger traffic was analyzed. Ship inter-arrival time was modeled by fitting Poisson distribution to historical data. Similarly, the service time was fit by Erlang distribution.

2.2 Ship Traffic Models for Simulation Studies

In this thesis the natural question is considered: given expected container vessel traffic, what is the most efficient quay partitioning? Thus, an STM is essential in simulation studies when a new container terminal is designed or an existing one is redesigned. In many port optimization formulations, quay partitioning into berths, the layout of the storage yard and the landside

container transport lanes must be chosen. These design decisions further impact future ship to berth assignments, tugboats and crane assignments, container storage and transport organization. Since container terminals are complex infrastructures which operations evolved over generations, there are many material, environmental, legal, financial constraints determining ship berths, crane schedules, vehicle logistics, container storage, and other port activities. Due to this complexity, it is hard to expect that analytical models will provide accurate estimations of the effects of the planned port changes. Furthermore, large time horizon of the future terminal use implies large uncertainty, which is difficult to take into account in analytical models. Therefore, simulation studies provide a natural approach to assessing future port performance.

Such studies require an accurate but versatile ship traffic model providing the right inputs for the simulation. In [5], 18 port simulation models built for risk and capacity assessment are reviewed. Most of the referred studies focus on nautical aspects, whereas terminal operations and ship arrival processes are represented only with some limited resolution. The extent of processes they depict demonstrates that modeling ship traffic is data-expensive and laborious. Hence, even partial models are helpful. They observed that uncertainty of arrival and processing times should be properly modeled and recommended to base the vessel arrival times on historical data, considering the stochasticity of the process. This is what is undertaken in this thesis.

Most current models of vessel arrival times use exponential or Poisson distributions [4, 2, 21, 76]. Service time was modeled using normal [4], Erlang [21, 76] distributions, or was given (deterministic) [2].

In [65] a statistical ship traffic model including vessel sizes, ship draft, number of TEUs, number of cranes and terminal revenue was developed on the basis of one port. The draft was calculated using linear regression on ship length, the number of cranes was estimated using linear regression on the number of transferred TEUs. The ship interarrival times were generated for all vessel sizes by one process with exponential distribution. Paper [2] considered three arrival processes: stock-controlled arrivals, equidistant arrivals, Poisson process. In the first two processes, ship arrival is planned but the actual time of arrival has three-point distribution with probabilities: 10% of arriving [2,12] hours before the expected time, 80% of arriving within ± 2 hours around the expected time, 10% of arriving [2,12] hours after the expected time.

In this thesis ship arrival and service times are represented using distributions derived from historical data available now thanks to AIS system [97]. A more diverse set, both with respect

to distribution types and their parameters, is used (see [90] and Chapter 5). In particular, the majority of ships arrive periodically with some dispersion of the return time, which is modeled as normal mixes dedicated to ship classes. Furthermore, we take into account the dependence of the return and service times on the vessel size class which was neglected in the earlier studies.

2.3 Berth Allocation Problem

Before proceeding to the BAP literature review, let us explain why BAP is as an essential element of this research. QPP is related to BAP because a QPP solution, i.e. a quay partition, is a BAP input. Moreover, the evaluation of a partition on one vessel arrival scenario equivalently solves a BAP instance. Several port throughput models are based on container streams with a tacit assumption of the arbitrary container volume divisibility. Scheduling theory [9, 17] indicates that such models may diverge from feasible schedules, because ships are indivisible. It has been shown in [17] that the ratio of the lengths of preemptive and nonpreemptive schedules on parallel identical processors is at most 4/3. In terms of BAP, this means that if re-berthing and suspension of ship processing are not allowed (which is a usual practice), then the best schedule can be approximately 33% longer than an optimistic approximation perceiving work as a divisible medium as in preemptive scheduling. When vessels are approaching and departing from the quay, the quay length may become *fragmented*, i.e. divided into many pieces too short to accommodate an arriving ship. A simplified perception assuming that TEU volume is a continuously divisible medium, allows for exploiting such fragmented quay space. In [9] contiguous and non-contiguous parallel task scheduling has been considered. The non-contiguous assignments allow flexible division of the ships into pieces distributed flexibly between fragmented free berth positions. Such assignments are obviously infeasible. The non-contiguous schedules may be shorter than the feasible contiguous ones. The worst-case ratio of the infeasible non-contiguous and feasible contiguous schedules lengths was bounded to the range of [125,200]% [9]. Thus, to obtain credible results on port capacity, solving BAP cannot be substituted by simpler approaches representing ship traffic as divisible container flows.

Studies on BAP are reviewed in [7, 8, 50, 71, 81]. BAP classes were distinguished on the basis of 1) quay layout, 2) vessel arrival type, 3) service time attributes and 4) performance measure [7]. Firstly, the quay layout types are: continuous, discrete, hybrid. In the continuous layout

vessels can be positioned anywhere along the quay. In the discrete case the quay is partitioned into berths such that at most one ship occupies a berth. Hybrid quay organization allows to assign more than one vessel in a berth. In this thesis we initially use discrete and hybrid layouts, such that in the hybrid case berths can be occupied by at most two ships, which prevents berth length fragmentation [9, 88]. After determining that discrete berth layout may be less robust than the hybrid layout (Chapter 3) we stick to the hybrid layout in the SQPP study (Chapter 6). Secondly, in the BAP formulations vessels arrive over time or all of them are ready at the same time. We assume that vessels arrive over time. Thirdly, service times can be fixed and known in advance, or may depend on ship berthing position, crane assignment and scheduling. In this thesis it is assumed that for a given scenario ship service times are known in advance and fixed, while in the SQPP case the set of scenarios has statistical features of real service times. Finally, a number of optimality criteria were considered in the past as the fourth defining attribute of BAP. The optimality criteria capture the goals of the port, terminal managers, and line shippers. For example, the turnaround time is the time a vessel spends waiting and mooring. The mean weighted turnaround time is equivalent to the mean weighted flow time used in the scheduling theory. Another frequently used criterion is the weighted sum of vessel tardiness. Other criteria are, e.g., the cost of container handling and leveling crane workload. We assume mean weighted flow time as the quality measure. According to the notation introduced in [7] the BAP version used in this thesis to evaluate quay partitions is denoted $hybr|dyn|fix|\sum weight$.

BAP can be analyzed on strategic, tactical and operational levels [7, 8, 62]. The levels differ in the considered time horizons and data uncertainty. At the strategic level, new container terminals are designed or decisions are made on expanding the capacity of the existing terminals in order to offer services to new shipping lines [37, 41]. At this stage, the data on ship traffic and handling resources may be very imprecise. The uncertainties can be tackled in simulations of alternative scenarios of the ship traffic and the handling processes. In the tactical BAP, time horizons of one or two weeks are studied. The input data are still uncertain. For instance, according to [86], over 40% of vessels are at least one day late. Though there are examples of stochastic approaches [74, 103], most papers consider discrete deterministic BAP with the assumption that vessel schedules can be adjusted at the operational level. Since ship service time depends on the resources allocation, berth allocation decisions are coupled with crane and container storage assignment, often resulting in multicriteria formulations [28, 38, 54, 56, 104,

Table 2.2: Selected BAP literature

Problem type	No. of vessels	No. of berths	year	ref.
BAP	50	10	2003	[34]
BAP	35	10	2005	[18]
BAP	200	20	2008	[35]
BAP	200	10	2010	[15]
BAP	60	13	2011-2014	[11, 51, 84]
BAP+QCSP	200	4	2017	[56]
terminal	110	16	2017	[75]
BAP	100	8	2018	[22]
BAP	40	2	2020	[33]
BAP	60	16	2021	[102]
BAP+CS	60	43	2021	[58]

QCSP – quay crane scheduling problem, terminal – whole terminal infrastructure modeled with BAP as a subproblem, CS – vessel sequencing in a navigation channel.

103, 105]. At the operational level, the time horizon of hours or at most a few days is considered and all data is assumed known and fixed. The goal of the operational planning is to stay as close as possible to the initial tactical level schedule. The methods developed in this thesis are dedicated to BAP at the strategic level, when decisions with time horizons of years, affecting thousands of ship arrivals, must be verified quickly.

In [71] authors review and propose taxonomy for BAP with uncertainty. Sources of uncertainty and methods of dealing with it can be the same both in BAP and in SQPP. According to the taxonomy of [71] we assume a proactive stochastic programming approach, with uncertain arrival and handling times represented with probability distributions which can be rendered as scenarios, with weighted waiting time as objective, hybrid quay allocation and we use heuristic methods. Let us note that beyond similarities with stochastic BAP there are also differences stemming from the change in time horizon. Stochastic BAP solution fixes berth allocation for vessels in the proactive phase, expecting variations in arrival and processing times. SQPP solution fixes the partition during the proactive phase, but not the berth allocation that can be recomputed later during the reactive or online phase. In a recent paper [72] consider the quay crane scheduling problem and propose a distributionally robust optimization (DRO) model which can generalize both stochastic programming (SP) and robust optimization. They conclude that SP and DRO approaches are preferable.

Since BAP is **NP**-hard in general [57], it can be solved to optimality only for small instances. Many advanced metaheuristics were used with success to solve BAP [7, 8]. A set of example

papers is collected in Tab.2.2 with a short name of the considered problem, numbers of ships and berths, year of publication, are given. Tab.2.2 asserts a gap between BAP sizes routinely solved in the existing literature, and the ones that have to be solved in our case. For example, the number of ship arrivals in 2016 ranged from 471 (in Gdańsk) to 18494 (Singapore) (see [90], Chapter 5). Due to the large number of ships for one year time horizon, the algorithm runtimes will be a key limitation in selecting methods solving BAP for a port simulation. It can be still argued that only short weekly patterns of arrivals suffice because ship arrivals are periodic. Indeed, 2-, 3-week return intervals seem natural, but 5-, 7-, 11-week return times are also quite common, especially on Asia-Europe, Europe-America lines ([90], Chapter 5). The least common multiple of these periods, necessary to grasp interactions between returns on different lines, easily spans a year. Furthermore, there might be large differences between the expected and the actual vessel arrival times, thus making periodic models less attractive [86, 90]. Consequently, the single short-time schedule pattern would inevitably diverge from the real traffic. Thus, the considered BAP instances are unavoidably large while the instances routinely solved in the existing literature are notably small. In order to avoid unacceptable computational complexity specialized algorithms must be selected.

Finally, let us note that the reality of port organization is much richer than that represented in BAP itself. BAP is tightly coupled with the quay crane assignment and scheduling, container storage allocation, inter-terminal transport planning, and hinterland communication. Consequently, BAP can be again a sub-problem in more general port optimization formulations. Such total approaches are beyond the scope of this thesis.

2.4 Algorithm Selection Problem

Choosing algorithms to evaluate a quay partition on a traffic scenario is subject of algorithm selection problem (ASP). Initially ASP was formalized by [70] as a problem of determining the best algorithm for a given instance of some problem given some performance measure. More formally, given: (i) set $\mathcal{I}$ of problem instances, called problem space, (ii) set $\mathcal{A}$ of algorithms, called algorithm space, and (iii) performance measure $p : \mathcal{I} \times \mathcal{A} \to R$, find a mapping $S : \mathcal{I} \to \mathcal{A}$ that maximizes performance $p(I, S(I))$ for each I. In an alternative formulation one more set is given: (iv) a set of instance characteristics $f(I)$, called feature space $\mathcal{F}$. Then, the algorithm

selection problem consists in finding a mapping $S : \mathcal{F} \rightarrow \mathcal{A}$ which maximizes performance $p(I, S(f(I)))$. In a typical setting, ASP is a machine learning problem of instance classification, that is, of determining the class (the algorithm) the instance belongs to. However, the above formulation is not the only one possible. There are various formalizations of ASP, with regard to the performance measures, when, how, and what algorithms to select. We recommend surveys [23, 49, 79, 80, 67] to the interested readers.

One approach to solving the ASP is to use algorithm portfolios. Using the algorithm portfolio minimizes the risk of an unfortunate choice of just one algorithm which may sometimes behave badly (see examples in Section 4.5). ASP becomes an optimization problem of covering a finite set $\mathcal{I}'$ of training instances in the most exhaustive way within the portfolio runtime limit [39, 44, 64]. Then, portfolio schedules determine the division of the runtime limit between the portfolio algorithms. In [64], the constraint satisfaction problem with $|\mathcal{A}| = 5$ algorithms is considered. Three types of schedules are tested: (1) split schedule giving each algorithm an equal amount of time, (2) static schedule giving each algorithm the amount of time which maximizes the number of instances covered in the entire problem space $\mathcal{I}'$, and (3) dynamic schedule built as in the static case but with a per-instance portfolio. The per-instance portfolio is constructed by first selecting a subset of $k = 10$ training instances closest to the considered instance, and then selecting algorithms which cover the greatest subset of the k instances within the runtime limit. The closest instances were selected using the Euclidean distance in the space of 36 features. Similarly, in [44] a per-instance algorithm portfolio is chosen dynamically as a subset of 37 boolean satisfiability (SAT) solvers. The distance between the current and the training instances is the Euclidean distance on 48 SAT instance features. In [39] timeout-optimal schedules are considered. Note that for certain training instances the algorithms may have runtimes exceeding the given limit, which results in the algorithm timeout. The portfolio has been selected and sequenced on the processors to minimize the number of instances not solved by any algorithm before the timeout. Further results on ASP and algorithm portfolios can be found in [23, 49, 79, 80, 67].

It can be observed that the existing methods either classify instances for the goal of finding the algorithm producing the best quality solution while ignoring the runtime cost, or solve search problems (SAT, constraint satisfaction) at a given time limit. Thus, to the best of our knowledge, most of the research on ASP concentrated either on the solution quality or

on the runtime as the performance indicators, but not both at the same time. It is a common expectation in combinatorial optimization (supported by Section 4.5 results) that fast algorithms provide worse solutions than other algorithms running in longer time. Thus, a runtime-quality trade-off exists and with changing the runtime limit the algorithms selected to a portfolio also change. It is commendable to inform the decision maker how the set of algorithms constructing the best solutions evolves with the runtime limit. Therefore, in this thesis we use the concept of algorithm portfolios covering a set of training instances with the best solutions which could be found in a given runtime limit, subject to the minimum computational cost of the chosen algorithms, introduced in [67, 88]. In particular we use results of [88] where algorithm portfolios for large BAP instances were considered (see Section 6.2). To evaluate a partition, a customized portfolio of algorithms effective both in solution quality and in time performance will be used.

Chapter 3

Deterministic Quay Partitioning Problem

3.1 Introducing Quay Partitioning Problem

In this chapter deterministic quay partitioning problem (DQPP) is considered. We assume that vessel arrival times, service times, lengths, and weights are given. Berth sizes are chosen from a finite set of possible lengths. The quality of quay partitioning is measured as vessel mean weighted flow time (MWFT). We assume that the quay can be partitioned according to one of two schemes: into berths that accommodate at most one ship in a berth, or at most two ships in a berth. The former scheme will be referred to as discrete layout [7, 8], or one ship in one berth, 1in1 for short. The latter will be called two ships in one berth (2in1), or a hybrid quay layout. The 2in1 scheme prevents berth length internal fragmentation (see [9, 75] and Section 2.3) because two ships can be always positioned at the opposite ends of one berth. The contributions of this chapter can be summarized in the following way: (1) Discrete quay partitioning problem is formulated and proved **NP**-hard. (2) New mixed integer linear formulations, for the two ways of handling vessels at the quay, are proposed and their scalability is tested. (3) Features of optimum solutions are analyzed, in particular: (i) sizes of chosen berths for changing vessel size mixture and under congestion; (ii) advantages (if any) of using a "one size fits all" approach; (iii) advantages of the two alternative ways (1in1 vs 2in1) of handling the

Table 3.1: Summary of notations for Chapter 3

B	set of admissible berth lengths, $B = \{\lambda_1, \ldots, \lambda_f\}$
c_j	ship j service completion time
f	number of admissible berth lengths, $f = \|B\|$
$F(K)$	mean weighted flow time for partition K
K	partition, vector of berth lengths frequencies, $K = (k_1, \ldots, k_f)$
k_i	number of berth length λ_i occurrences in a partition
λ_i	admissible berth length i
L_j	length of ship j
L_{max}	longest ship length, $L_{max} = \max_{i=1}^{n}\{L_i\}$
m	number of berths in K, $m = \sum_{i=1}^{f} k_i$
n	number of vessels, $n = \|V\|$
p_j	service time of ship j
Q	quay length
ρ_i	maximum number of berths of the ith length, $\rho_i = \lfloor Q/\lambda_i \rfloor$
r_j	ship j ready time
V	set of vessels
w_j	weight (value) of servicing ship j

vessels. (4) This allowed to give recommendations for quay partitioning into berths.

Further organization of this chapter is the following. In Section 3.2 DQPP is formulated. The MIPs are presented in Sections 3.3 and 3.4. The time performance of the two MIPs is analyzed in Section 3.5. Section 3.6 is dedicated to a study of the DQPP solutions features. We summarize the results in Section 3.7. The notations used in the current chapter are collected in Tab. 3.1.

3.2 Problem Formulation

We assume that a quay of length Q is to be partitioned into berths of lengths given in set $B = \{\lambda_1, \ldots, \lambda_f\}$. Without loss of generality we assume $\lambda_1 < \cdots < \lambda_f$. We will be saying that berths of equal length have the same type. Arriving vessels are given in set V and $|V| = n$. Each arriving vessel j is defined by its arrival time r_j, length L_j, service time (alternatively called processing time) p_j, and weight w_j. Weight of the ship represents importance of the ship arrival, e.g. cost of the service. Though the overwhelming majority of ships, as physical objects, are returning to a container terminal (Chapter 5), the distinction between a ship (as a physical object) and one of its arrivals at a terminal is immaterial in the context of DQPP. Therefore, terms ship, vessel and arrival will be used exchangeably.

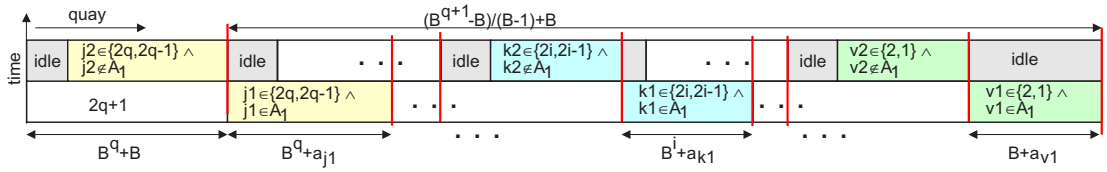


Figure 3.1: Partition and schedule for **NP**-hardness proof of Theorem 1.

A solution to a DQPP is a partition of quay length Q into berths which can be represented as vector $K = (k_1, \ldots, k_f)$ of berth lengths frequencies. By definition $\sum_{i=1}^{f} \lambda_i k_i \leq Q$ and $L_{max} = \max_{j=1}^{n}\{L_j\} \leq \max_{i=1}^{f}\{\lambda_i : k_i > 0\}$, i.e., for all ships there are berths at which the ships can be feasibly positioned. The total number of berths in some DQPP solution is $m = \sum_{i=1}^{f} k_i$. Given some partition K, the ships in V have to be scheduled on the berth for minimum weighted flow time. Let c_j denote completion time of ship j service. The objective function assessing quality of some partition K is mean weighted flow time (MWFT): $F(K) = \sum_{j=1}^{n} w_j(c_j - r_j)/\sum_{j=1}^{n} w_j$. The above description can be summarized in the following definition:

Definition 1 (Discrete Quay Partitioning Problem). Given admissible berth lengths in set B, quay length Q and a set of vessels V, find partition K and a schedule for the vessels with minimum $MWFT$.

Let us observe that 2in1 berthing scheme generalizes 1in1 because schedules with 1 ship in a berth are feasible schedules in 2in1. A further advantage of 2in1 is that it gives more flexibility than 1in1 (because two ships i, j satisfying $L_i + L_j \leq \lambda_h$ can be served at the same time at some berth h) while simultaneously preventing internal fragmentation of the berth length as explained in the Introduction.

The assignment of the vessels from V in time alone, i.e. for fixed K, is a berth allocation problem (BAP) which is **NP**-hard [57]. Hence, DQPP is **NP**-hard even for fixed K. In the following we prove that also given the flexibility of choosing layout K, DQPP remains **NP**-hard. The proof applies both in 1in1 and 2in1 cases.

Theorem 1. *DQPP is* **NP***-hard.*

Proof. Let $\rho' = \lceil Q/\lambda_1 \rceil$ denote an upper bound on the number of berths of any length. Firstly, DQPP is in class **NP**. It suffices that generating module of NDTM guesses partition K and vessel schedule. The first can be written in polynomial time $O(f \log \rho')$. The schedule can be guessed

as pairs for each vessel: the assigned berth, position in the berth vessel sequence. The schedule can be written, its feasibility checked, and MWFT calculated in $O(n(\log n + \log f + \log \rho'))$. Secondly, a polynomial-time transformation from EVEN-ODD PARTITION will be shown.

EVEN-ODD PARTITION: given a set $A = \{a_1, \ldots, a_{2q}\}$ of positive integers, is there a subset $A_1 \subset A$ such that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A \setminus A_1} a_i = B$ and A_1 contains exactly one element from pair a_{2i-1}, a_{2i}, for $i = 1, \ldots, q$?

For brevity let us denote $\sum_{j=1}^{q} B^i = (B^{q+1} - B)/(B-1)$ as Z. An instance of DQPP is defined as follows: $Q = B^q + Z + 2B, f = 2q+1, \lambda_{2i-1} = B^i + a_{2i-1}, \lambda_{2i} = B^i + a_{2i}$, for $i = 1, \ldots, q$, $\lambda_{2q+1} = B^q + B$, $n = 2q+1, L_{2i-1} = w_{2i-1} = B^i + a_{2i-1}, L_{2i} = w_{2i} = B^i + a_{2i}$, for $i = 1, \ldots, q$, $L_{2q+1} = B^q + B, w_{2q+1} = B^{4q}$, and $r_i = 0, p_i = 1$, for $i = 1, \ldots, 2q+1$. We ask if a partition K and schedule exist such that $MWFT' = MWFT * W \leq B^{4q} + 3Z + 3B$, where $W = \sum_{j=1}^{2q+1} w_j$ is constant.

Suppose the answer to EVEN-ODD PARTITION is positive and set A_1 exists, then a partition and the schedule are shown in Fig.3.1. The quay is partitioned into $q+1$ berths such that no free space remains. The schedule has length of two time units. A berth of lengths $B^q + B$ hosts vessel $2q+1$ in the first time unit and the vessel from pair $\{2q, 2q-1\}$ *not* chosen to A_1 in the second time unit. Let this berth have index $q+1$. A berth with index $i = q, \ldots, 1$ of length $B^i + a_j$ is created, where a_j is the element from pair $\{2i, 2i-1\}$ chosen to A_1. Of the two vessels $\{2i, 2i-1\}$, the one corresponding with the element j chosen to A_1 is scheduled in the first time unit on berth i and the second vessel $\{2i, 2i-1\} \setminus \{j\}$ is scheduled in the second time unit on berth $i+1$ (the next longer one). The total occupied quay length in the first time unit is $B^q + B + \sum_{i=1}^{q} \sum_{j \in \{2i, 2i-1\} \cap A_1} (B^i + a_j) = B^q + B + Z + B = Q$, while $MWFT' = B^{4q} + \sum_{i=1}^{q} B^i + \sum_{a_j \in A_1} a_j + 2\sum_{i=1}^{q} B^i + 2\sum_{a_j \in (A \setminus A_1)} a_j = B^{4q} + 3Z + 3B$.

Suppose the answer to DQPP is positive, we will show that the answer to EVEN-ODD PARTITION must be also positive. Due to its weight $w_{2q+1} = B^{4q}$, vessel $2q+1$ must be scheduled in the first time unit, there must be also a berth of length at least $B^q + B$ for this vessel. Hence, at most $Z + B$ quay length remains for other berths. Since $Z + B < 2B^q$, for sufficiently large B, at most one berth of length λ_{2q} or λ_{2q-1} can be created in the remaining area. Suppose fewer than one berth of lengths λ_{2q} or λ_{2q-1} is created. Since L_{2q} and L_{2q-1} are greater than any $\lambda_{2i}, \lambda_{2i-1}$ for $i < q$, vessels $2q, 2q-1$ have to be scheduled in the second and third time units on the berth of

length $B^q + B$ after vessel $2q+1$. In such a case $MWFT' \geq B^{4q} + 2B^q + 2a_{2q} + 3B^q + 3a_{2q-1} > B^{4q} + 5B^q > B^{4q} + 3(B^{q+1} - B)/(B-1) + 3B$ for sufficiently large B. Hence, exactly one berth of one of the lengths λ_{2q} or λ_{2q-1} has to be created. Let us observe that $MWFT'$ as defined in the question of this transformation is calculated with the assumption that one vessel from each pair $\{2i, 2i-1\}$ is scheduled in the first time unit and the other in the second time unit. Suppose none of the two vessels is scheduled in the first time unit. Thus, at most 2 shorter vessels $i \in \{2k, 2k-1\}, j \in \{2\ell, 2\ell-1\}$ (for 2in1, for 1in1 only one vessel) can exploit the berth of length λ_{2q} or λ_{2q-1}. By processing one of the vessels $2q, 2q-1$ in the second time unit, or later, instead of processing it in the first time unit, the $MWFT'$ increases by at least $B^q + a_{h \cap \{2q, 2q-1\}}$ while it decreases by at most $2B^{\max\{k,\ell\}} + a_i + a_j < B^q + a_{h \cap \{2q, 2q-1\}}$, for sufficiently large B, because $q > \max\{k, \ell\}$. As a consequence, a strictly better solution is obtained by using this berth for one of the two vessels $2q, 2q-1$ in the first time slot. Otherwise the value of $MWFT'$ cannot be met. These observations can be inductively extended to all pairs of berth lengths $\{\lambda_{2i}, \lambda_{2i-1}\}$, for $i = q-1, \ldots, 1$, and one vessel from each pair $\{2i, 2i-1\}$ is scheduled in the first time unit. Let E_1, E_2, E_3 be the set of vessels, except for $2q+1$, scheduled in the first, second and third (and possibly later) time units. Let $W_1 = \sum_{j \in E_1} w_j$, $W_2 = \sum_{j \in E_2} w_j$, $W_3 = \sum_{j \in E_3} w_3$. The vessels in E_1 can occupy quay length at most equal to the sum of lengths of the chosen berths, which is bounded by the remaining quay length. So $\sum_{i=1}^{q} \sum_{j \in \{2i, 2i-1\} \cap E_1} L_j = \sum_{i=1}^{q} B^i + \sum_{j \in E_1} a_j \leq Z + B$, and hence $\sum_{j \in E_1} a_j \leq B$. The $MWFT'$ of vessels in E_1, E_2, E_3 is $MWFT' = 3(\sum_{=1}^{2q} w_j - W_1 - W_2) + 2W_2 + W_1 = 6Z + 6B - W_2 - 2W_1 \leq 3Z + 3B$ because a schedule with the required $MWFT'$ exists. Since $W_1 = Z + \sum_{j \in E_1} a_j, W_2 \leq Z + \sum_{j \in A \setminus E_1} a_j$ the above condition is satisfied only if $W_1 \geq Z + \sum_{j \in E_1} a_j \geq B$. Consequently, set A_1 satisfying condition $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A \setminus A_1} a_i = B$ must exist. Note that from each pair $\{2i, 2i-1\}$ of berth lengths only one is created. Analogously, from each pair $\{2i, 2i-1\}$ of vessels only one is scheduled in the first time unit. Hence, from each pair $\{a_{2i}, a_{2i-1}\} \subset A$ only one is selected to E_1, and A_1 must exist. □

Thus, DQPP is **NP**-hard already in quite restricted setting, i.e., even if all vessels have unit processing time and are simultaneously available ($\forall_j p_j = 1, r_j = 0$).

3.3 Mixed Integer Linear Program for 1in1

The MIP formulation for 1in1 extends [11, 18]. Yet, since a qualitatively different problem is solved, not only new constraints had to be added, but all constraints were changed to accommodate the options for using one of many possible realizations of berths. Let us give some intuition before proceeding to the MIP itself. A similarity to vehicle routing problem (VRP) is used. A berth corresponds here to a truck, a ship is an equivalent of a customer. A sequence of ships on a berth can be treated like a sequence of customers visited by one truck in VRP. The MIP uses copies of the berths of certain lengths. Each such copy (like a truck) has its sequence of visiting ships. The MIP must select the number of copies of a certain berth length such that their total length does not exceed quay length Q. In the MIPs we will use the following notations:

Constants:

o – a dummy origin ship that precedes all other ships in all berth ship sequences.

d – a dummy destination ship that follows all other ships on all berths.

$\rho_i = \lfloor Q/\lambda_i \rfloor$ - maximum number of type i berths, for $i = 1, \ldots, f$.

M – set of all possible berths. An element of M is a couple (k, ℓ) with $k \in \{1, \ldots, f\}$ and $\ell \in \{1, \ldots, \rho_k\}$.

$M'_j \subseteq M$ – all possible berths incompatible with vessel j, i.e. all berths i and their ρ_i copies satisfying $\lambda_i < L_j$.

G – a large number, for example $G > \sum_{j=1}^{n} p_j + \max_{j=1}^{n}\{r_j\}$.

Decision variables:

$t_j^{k\ell}$ – start of processing ship j on berth type k copy ℓ.

$x_{ij}^{k\ell} = 1$ if ship i immediately precedes ship j on berth type k copy ℓ, 0 otherwise.

$y^{k\ell} = 1$ if berth type k copy ℓ is in use, 0 otherwise.

A mixed integer linear program for 1in1 is as follows:

MIP DQPP-1in1:

$$\min \sum_{i \in V} w_i \sum_{(k,\ell) \in M} \left(t_i^{k\ell} - r_i + p_i \sum_{j \in V \cup \{d\}} x_{ij}^{k\ell} \right) \tag{3.1}$$

s.t.

$$\sum_{(k,\ell)\in M}\sum_{j\in V\cup\{d\}} x_{ij}^{k\ell} = 1 \qquad i = 1,\dots,n \tag{3.2}$$

$$\sum_{j\in V\cup\{d\}} x_{oj}^{k\ell} = 1 \qquad (k,\ell)\in M \tag{3.3}$$

$$\sum_{i\in V\cup\{o\}} x_{id}^{k\ell} = 1 \qquad (k,\ell)\in M \tag{3.4}$$

$$\sum_{j\in V\cup\{d\}} x_{ij}^{k\ell} = \sum_{j\in V\cup\{o\}} x_{ji}^{k\ell} \qquad (k,\ell)\in M, i\in V \tag{3.5}$$

$$t_i^{k\ell} + p_i - t_j^{k\ell} \le (1 - x_{ij}^{k\ell})G \qquad \forall (k,\ell)\in M, (i,j)\in V\times V \tag{3.6}$$

$$t_i^{k\ell} \ge r_i \qquad \forall (k,\ell)\in M, i\in V \tag{3.7}$$

$$y^{k\ell} \ge x_{ij}^{k\ell} \qquad \forall (k,\ell)\in M, i\in V\cup\{o\}, j\in V \tag{3.8}$$

$$y^{k\ell} \ge y^{k,\ell+1} \qquad \forall k, \ell = 1,\dots,\rho_k - 1 \tag{3.9}$$

$$\sum_{k=1}^{f}\sum_{\ell=1}^{\rho_k} y^{k\ell}\lambda_k \le Q \tag{3.10}$$

$$x_{ij}^{k\ell} = 0 \qquad \forall k\in M_i'\cup M_j', i,j\in V\cup\{o,d\}, \ \ell = 1,\dots,\rho_k \tag{3.11}$$

$$y^{k\ell}\in\{0,1\} \qquad \forall (k,\ell)\in M \tag{3.12}$$

$$x_{ij}^{k\ell}\in\{0,1\} \qquad \forall (k,\ell)\in M, (i,j)\in (V\cup\{o,d\})\times(V\cup\{o,d\}) \tag{3.13}$$

$$t_i^{k\ell}\in\mathcal{Q}^+ \qquad \forall (k,\ell)\in M, i\in V \tag{3.14}$$

In the above formulation the objective function (3.1) minimizes weighted sum of vessel completion times. By (3.2) each vessel is served at some berth. Equations (3.3), (3.4) ensure that each berth has a beginning, and respectively, an end of the chain of served vessels. Constraint (3.5) guarantees that a ship on any berth has as many predecessors as successors. This, together with constraints (3.3), (3.4), ensures that there is a chain of ships at the berth and only one ship at the berth at a time. The service starting time of ship j immediately succeeding ship i on berth k copy ℓ does not overlap the service time of i by inequality (3.6). According to constraint (3.7) a vessel is processed after arriving. Inequality (3.8) ensures that vessels i, j can use k-th berth type copy ℓ only if this copy is allowed to be used. By (3.9), copy $\ell+1$ of berth type k can be used only if copy ℓ is already in use. Inequality (3.10) ensures that all used copies

of different berth types fit in the quay length. Constraints (3.11) guarantee that ships i, j are assigned only to the berth types k such that $L_i \leq \lambda_k$ and $L_j \leq \lambda_k$. $\mathcal{Q}^+$ in (3.14) means that ship service starting times are positive fractional numbers.

3.4 Mixed Integer Linear Program for 2in1

When there can be at most two vessels in one berth, then each schedule of the vessels at the berth can be transformed such that the vessels are mooring either at the left or at the right end of the berth [88]. Then the chain of vessels assigned in each end of a berth can be treated as equivalent to a truck route between customers. Hence, we will use the same vehicle routing idea as in the 1in1 case, but we will refer to the left and the right chain at each berth. Two ships i, j simultaneously sharing berth of type k must satisfy $\lambda_k \geq L_i + L_j$. However, it is feasible to assign to a berth ships violating this constraint on the condition that they are not serviced simultaneously. We will call such a situation an interference of ships i, j. Consequently, MIP (3.1)-(3.14) must be extended to handle also ship interferences. Let us introduce additional notations:

Constants:

$N^k = \{(i,j) : L_i \leq \lambda_k, L_j \leq \lambda_k, L_i + L_j > \lambda_k\}$ - a set of vessel i, j pairs which cannot be executed simultaneously on berth type k.

$Y^k = \{(i,j) : L_i + L_j \leq \lambda_k\}$ - a set of vessel i, j pairs which can be executed simultaneously on berth type k.

Decision variables:

$s_{ij}^{k\ell} = 1$ if vessels i, j share berth k copy ℓ, are in different (left, right) chains, and their total length exceeds berth length (i.e. $L_i \leq \lambda_k, L_j \leq \lambda_k, L_i + L_j > \lambda_k$ hence i, j interfere), 0 otherwise.

$z_{ij}^{k\ell} = 1$ if vessels i, j share berth k copy ℓ, are in different (left, right) chains, their total length exceeds berth length, and vessel i is executed before vessel j, 0 otherwise.

$xL_{ij}^{k\ell} = 1$ if vessel i immediately precedes vessel j in the left chain of berth k copy ℓ, 0 otherwise.

$xR_{ij}^{k\ell} = 1$ if vessel i immediately precedes vessel j in the right chain of berth k copy ℓ, 0 otherwise.

MIP DQPP-2in1:

$$\min \sum_{i\in V} \sum_{(k,\ell)\in M_i} w_i \left(t_j^{k\ell} - r_i + p_i \sum_{j\in V\cup\{d\}} (xL_{ij}^{k\ell} + xR_{ij}^{k\ell}) \right) \tag{3.15}$$

s.t.

$$z_{ij}^{k\ell} + z_{ji}^{k\ell} = s_{ij}^{k\ell} \qquad \forall (k,\ell) \in M, (i,j) \in N^k \tag{3.16}$$

$$s_{ij}^{k\ell} = 0 \qquad \forall (k,\ell) \in M, \{i,j\} \in Y^k \tag{3.17}$$

$$t_j^{k\ell} \geq t_i^{k\ell} + p_i - G(2 - z_{ij}^{k\ell} - s_{ij}^{k\ell}) \qquad \forall (k,\ell) \in M, (i,j) \in N^k \tag{3.18}$$

$$t_i^{k\ell} \geq t_j^{k\ell} + p_j - G(1 + z_{ij}^{k\ell} - s_{ij}^{k\ell}) \qquad \forall (k,\ell) \in M, (i,j) \in N^k \tag{3.19}$$

$$\sum_{i'\in V\cup\{d\}} xL_{ii'}^{k\ell} + \sum_{i'\in V\cup\{d\}} xR_{ji'}^{k\ell} - 1 \leq s_{ij}^{k\ell} \qquad \forall (k,\ell) \in M, (i,j) \in N^k \tag{3.20}$$

$$\sum_{i'\in V\cup\{d\}} xR_{ii'}^{k\ell} + \sum_{i'\in V\cup\{d\}} xL_{ji'}^{k\ell} - 1 \leq s_{ij}^{k\ell} \qquad \forall (k,\ell) \in M, (i,j) \in N^k \tag{3.21}$$

$$z_{ij}^{k\ell}, s_{ij}^{k\ell} \in \{0,1\} \qquad (k,\ell) \in M, i,j \in V \tag{3.22}$$

$$\sum_{(k,\ell)\in M} \sum_{j\in V\cup\{d\}} (xL_{ij}^{k\ell} + xR_{ij}^{k\ell}) = 1 \qquad i = 1,\ldots,n \tag{3.23}$$

$$\sum_{j\in V\cup\{d\}} xL_{oj}^{k\ell} = 1, \sum_{j\in V\cup\{d\}} xR_{oj}^{k\ell} = 1 \qquad (k,\ell) \in M \tag{3.24}$$

$$\sum_{i\in V\cup\{o\}} xL_{id}^{k\ell} = 1, \sum_{i\in V\cup\{o\}} xR_{id}^{k\ell} = 1 \qquad (k,\ell) \in M \tag{3.25}$$

$$\sum_{j\in V\cup\{d\}} xL_{ij}^{k\ell} = \sum_{j\in V\cup\{o\}} xL_{ji}^{k\ell} \qquad (k,\ell) \in M, i \in V \tag{3.26}$$

$$\sum_{j\in V\cup\{d\}} xR_{ij}^{k\ell} = \sum_{j\in V\cup\{o\}} xR_{ji}^{k\ell} \qquad (k,\ell) \in M, i \in V \tag{3.27}$$

$$t_i^{k\ell} + p_i - t_j^{k\ell} \leq (1 - xL_{ij}^{k\ell})G \qquad \forall (k,\ell) \in M, (i,j) \in V \times V \tag{3.28}$$

$$t_i^{k\ell} + p_i - t_j^{k\ell} \leq (1 - xR_{ij}^{k\ell})G \qquad \forall (k,\ell) \in M, (i,j) \in V \times V \tag{3.29}$$

$$t_i^{k\ell} \geq r_i \qquad \forall (k,\ell) \in M, i \in V \tag{3.30}$$

$$y^{k\ell} \geq xL_{ij}^{k\ell} \qquad \forall (k,\ell) \in M, i,j \in V \tag{3.31}$$

$$y^{k\ell} \geq xR_{ij}^{k\ell} \qquad \forall (k,\ell) \in M, i,j \in V \tag{3.32}$$

$$y^{k\ell} \geq y^{k,\ell+1} \qquad \forall k, \ell = 1, \ldots, \rho_k - 1 \tag{3.33}$$

$$\sum_{k=1}^{f} \sum_{\ell=1}^{\rho_k} y^{k\ell} \lambda_k \leq Q \tag{3.34}$$

$$xL_{ij}^{k\ell} = 0, \quad xR_{ij}^{k\ell} = 0 \qquad \forall k \in M_i' \cup M_j' \tag{3.35}$$

$$y^{k\ell} \in \{0,1\} \qquad \forall (k,\ell) \in M \tag{3.36}$$

$$x_{ij}^{k\ell} \in \{0,1\} \qquad \forall (k,\ell) \in M, \quad (i,j) \in (V \cup \{o,d\}) \times (V \cup \{o,d\}) \tag{3.37}$$

$$t_i^{k\ell} \in \mathcal{Q}^+ \qquad \forall (k,\ell) \in M, i \in V \tag{3.38}$$

By (3.16) two ships in left and right chains on berth type k copy ℓ (then $s_{ij}^{k\ell}$=1) that cannot be executed simultaneously must be sequenced such that i precedes j, or vice versa. When two vessels i, j are short enough to use berth type k simultaneously but in different chains, then by (3.17) the mutual sequence of the two vessels in different chains of berth type k copy ℓ need not be set. Inequalities (3.18),(3.19) ensure that service times of two vessels in the same berth type k copy ℓ, but in two different chains (left, right) do not overlap if their total length is a greater than the berth length. Observe that for $s_{ij}^{k\ell} = 0$ both constraints are inactive. For $s_{ij}^{k\ell} = 1$ also exactly one of $z_{ij}^{k\ell}, z_{ji}^{k\ell}$ must be 1 by (3.16). Consequently, for $s_{ij}^{k\ell} = 1$, one of inequalities (3.18),(3.19) is binding and one is inactive and interfering ships i, j processing intervals do not overlap. Constraints (3.20), (3.21) guarantee that vessels i, j processed on the same berth type k copy ℓ, in different chains (left, right), which cannot be serviced at the same time have their service times separated by setting $s_{ij}^{k\ell} = 1$. Constraints (3.23)–(3.38) are analogous to (3.2)-(3.14), but now they are applied separately to two chains (left/right) in the same berth. Precisely, (3.23) ensure that a ship is present in one chain only. By (3.24), (3.25) each chain of ships serviced at berth k copy ℓ has a beginning and an end. Conditions (3.26), (3.27) are flow conservation constraints for each ship i and berth type k copy ℓ. Vessels assigned to a chain (left or right) on berth k, ℓ are processed sequentially by (3.28), (3.29). Ships cannot be assigned to a left or right chain on berth k copy ℓ if this berth type and the copy is not in use by inequalities (3.31), (3.32). The remaining constraints are the same as in (3.2)–(3.14).

3.5 Scalability of the Methods

In this section we report on time performance of solving the MIPs to optimality and the main determinants of instance hardness. All the tests were conducted on a PC computer with Intel i7-8550U CPU@2GHz, 32GBRAM and Windows 10. The MIPs were solved by CPLEX 12.8 with standard settings and runtime limit of 10hours. Unless stated to be otherwise, quay length was set to $Q = 20$, the number of different berth lengths was set to $f = 4$. Particular berth lengths $\lambda_1, \ldots, \lambda_f$ were chosen from $U[1, 10]$, that is from a discrete uniform distribution in range [1,10]. Depending on the test setting, the number of vessels n was one number from range $[5, 11]$. Ship lengths L_j were chosen from $U[1, \max_{i=1}^{f}\{\lambda_i\}]$. Ship service times were generated from $U[1, 10]$. Arrival times were generated as $r_1 = 0, r_{j+1} = r_j + U[0, 2]$, for $j = 2, \ldots, n$. These units can represent real vessels and service times when appropriately scaled, e.g., our unit of length can be 40m and processing time unit can be 3 hours. Ship weights were chosen according to $w_j = 0.5 \times L_j p_j + U[0, 20]$, for $j = 2, \ldots, n$. The dispersion of ship parameters in the test instances also serves the purpose of modeling uncertainty in ship traffic.

Execution times are shown in Fig.3.2a, c, e for 2in1 and Fig.3.2b, d, f for 1in1 cases, respectively. 50 instances were generated and solved for each point shown in the following Fig.3.2. In total over 1950 instances were solved to make Fig.3.2. All test datasets used to this end are available from [94]. Boxplots with whiskers show quartiles of the runtimes in a population of 50 instances. Each picture includes a median runtime trend line (marked "Median Aprox") with its equation and coefficient of determination R^2. The trend lines were chosen among linear, quadratic, exponential, power functions with maximum R^2. Finally, the greatest time and median time in the 50 instances population when the final solution was found are shown as red and green lines, respectively, marked as "Last Updt". It can be seen in Fig.3.2a and Fig.3.2b that execution time grows exponentially with the number of arrivals n. In the 2in1 model, runtime for $n = 11$ was longer than 10 hours limit in more than 25% of instances. Since such tests were interrupted, the results would be biased. For this reason the point for $n = 11$ is not shown in Fig.3.2a.

Judging by the coefficient of determination R^2 of the exponential median runtime trend line, n is an important determinant of the two MIPs runtimes. Execution times grow also with the number of admissible berth lengths f (see Fig.3.2c and Fig.3.2d). Yet, the growth intensity is

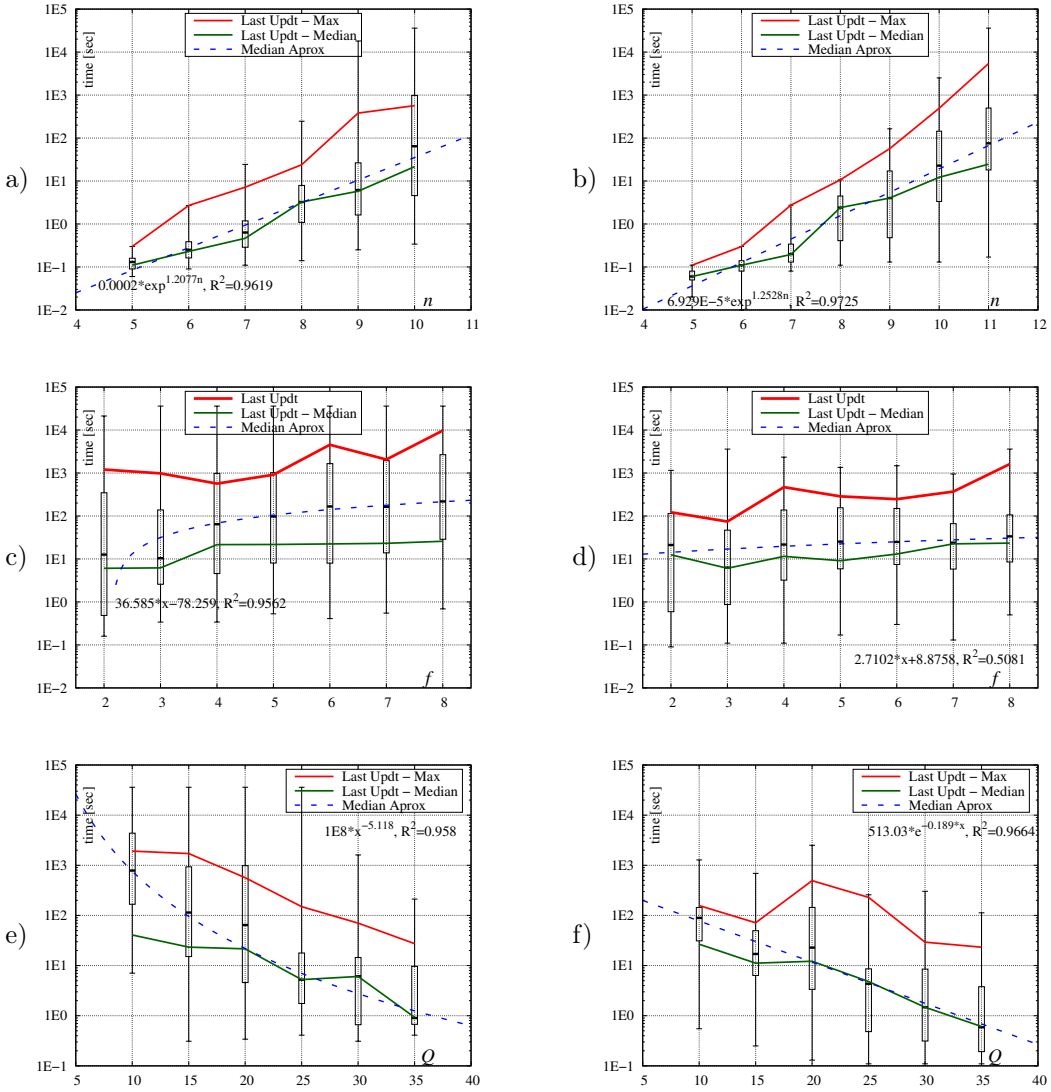


Figure 3.2: MIP runtime a) 2in1 vs n, b) 1in1 vs n, c) 2in1 vs f, d) 1in1 vs f, e) 2in1 vs Q, f) 1in1 vs Q.

Table 3.2: Coefficient of correlation between the logarithm of the runtime and instance congestion indicators

1in1	$n = 10, f = 4$	$n = 10, f = 8$	$n = 8, f = 4$
$\alpha = \sum_{j=1}^{n} p_j L_j / (Q \max_{j=1}^{n}\{r_j + p_j\})$	0.82	0.89	0.72
$\tau = \max_{k,\ell,,j}\{t_j^{k\ell} - r_j\}$	0.89	0.79	0.81
2in1	$n = 10, f = 4$	$n = 10, f = 8$	$n = 8, f = 4$
$\alpha = \sum_{j=1}^{n} p_j L_j / (Q \max_{j=1}^{n}\{r_j + p_j\})$	0.82	0.81	0.81
$\tau = \max_{k,\ell,,j}\{t_j^{k\ell} - r_j\}$	0.87	0.60	0.89

less strong because the linear trend line was the best fit. In the case of 1in1 (Fig.3.2d) f is a weak predictor of the runtime, because $R^2 \approx 0.5$ is not big. It can be seen Fig.3.2e and Fig.3.2f that with growing quay length Q instances are easier to solve. This is intuitively expected because increasing Q, while keeping the size of arrivals set constant, over-provisions the area to serve the vessels. Results in Fig.3.2e and Fig.3.2f correspond with the results in Tab.3.2 discussed below.

The strength of the relationship between the runtime and cardinality of set M was also analyzed, and it was even lower because coefficients of determination even for best fitting trend lines were below $R^2 = 0.06$. We analyzed the correlation between the runtime and two instance time congestion indicators: area ratio calculated as $\alpha = \sum_{j=1}^{n} p_j L_j / (Q \max_{j=1}^{n}\{r_j + p_j\})$ and instance time tightness $\tau = \max_{k,\ell,j}\{t_j^{k\ell} - r_j\}$. Area ratio α is the ratio of time-space area $Q \max_{j=1}^{n}\{r_j + p_j\}$ available until the earliest possible finish time of vessel service to the vessel demand for service in time and space $\sum_{j=1}^{n} p_j L_j$. With the increasing values of α the time-space area is increasingly over-subscribed. Time tightness τ is ship longest wait time from its arrival (r_j) until its processing starts ($t_j^{k\ell}$). Example correlations between the two indicators and logarithm of the runtime are collected in Tab.3.2. A rule of thumb stating that correlation is strong if the coefficient of correlation is greater than 0.7 is frequently satisfied. Thus, the two indicators can be used to predict runtime reliably.

Consider now when the solution values were updated for the last time. Let us remind that there are lines for the median and the largest last update times in Fig.3.2, denoted "Last Updt". Position of the line showing median time when the solution quality was updated for the last time (green line in Fig.3.2), shows that very often the optimum solution is found quite early and in a great part of the execution time proof of optimality is developed. Considering the position of the (red) line showing the maximum time in a population of 50 instances when the solution quality was updated for the last time, we decided to limit runtime in the following tests to one

Table 3.3: Average optimality gaps and ratios of the last delivered solution $MWFT$ to the first found solution $MWFT$, in 10 large random instances at 10-hour runtime limit

1in1, n	25	50	75	100
avg.opt. gap %	52.63	70.88	80.67	89.40
avg.last/first $MWFT$	0.31	0.28	0.27	0.39
No-solutions	0	0	0	0
2in1, n	25	50	75	100
avg.opt. gap %	47.73	71.96	86.97	94.60
avg.last/first $MWFT$	0.27	0.26	0.23	0.59
No-solutions	0	0	3	4

Table 3.4: Le Havre DQPP examples solved by MIP DQPP-1in1 and DQPP-2in1

instances*	w1	w2	w3	w4	w5	fo1	fo2	fo3	fo4	fo5
n	42	48	45	47	46	85	86	80	97	83
					1in1					
$MWFT$	6567657	9998557	7406544	6693338	6287105	13229317	18476791	10391603	22162648	10396155
opt.gap† %	0	0	0	0	0	19.81	5.45	0	41.35	0.95
					2in1					
$MWFT$	6567657	10041292	7406544	6750758	6400477	121859861	126515016	10391603	126367195	10297593
opt.gap† %	0	0.43	0	0.85	1.77	91.29	86.19	0	89.71	0

*) w1-w5 – one week examples, fo1-fo5 - fortnight examples; † opt.gap=0 – solution is optimum

hour. Among the instances solved for Fig.3.2 only in 4 cases were the solutions updated after 1 hour runtime.

The runtime ratios of MIPs solving the same instance according to the 2in1 and 1in1 models depend very much on the particular instance. Yet, in the tests with changing n (Fig.3.2a,b), f (Fig.3.2c,d), Q (Fig.3.2e,f), the runtimes of 2in1 MIPs were on average 2.2, 4.1, 4.0 times longer than for 1in1 MIPs, respectively.

The above results may seem pessimistic considering solvable instance sizes. Yet, MIP solvers can be used as heuristics if limiting their runtime or widening the optimality gap. In Tab.3.3 we report on 10-hour solvability of large random instances generated as described above. This approach allows to solve weekly and fortnight arrivals patterns in large European container terminals such as Le Havre. Tab.3.4 comprises $MWFT$s and optimality gaps obtained for exemplary week and two week arrivals in Le Havre. It can be noted that the results for these practical instances are notably better than the results for random instances, especially for 1in1 model.

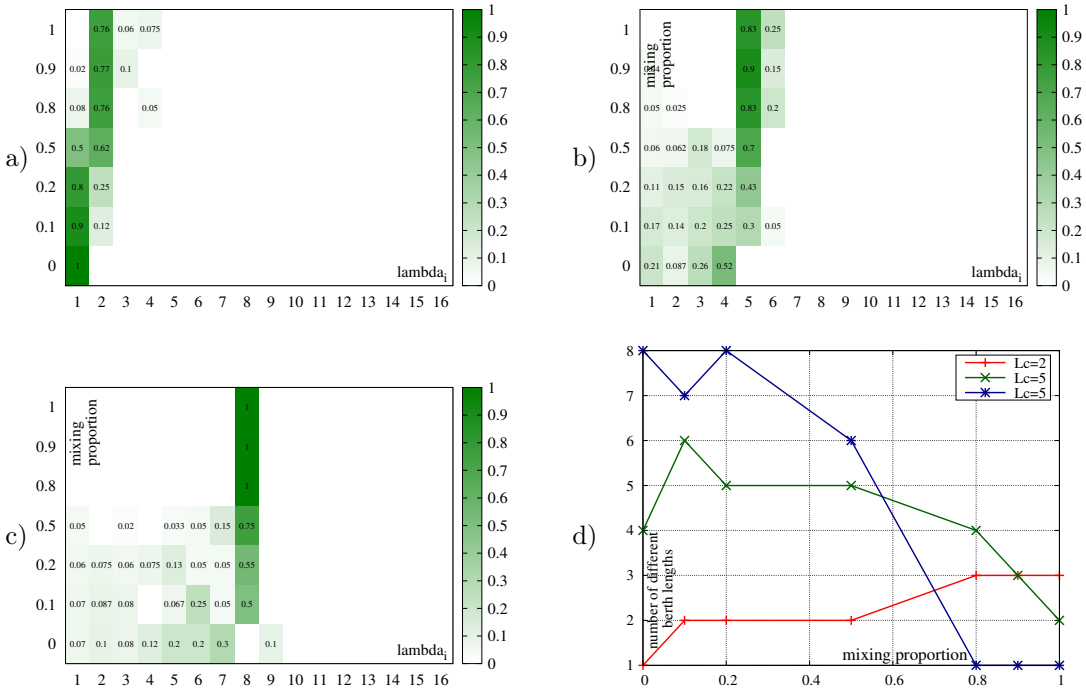


Figure 3.3: Berth lengths used in the solutions for 1in1. a) $L_{max} = 2$, b) $L_{max} = 5$, c) $L_{max} = 8$, d) Number of different berth lengths.

3.6 Features of DQPP Solutions

In this section we experimentally study features of the optimum, or near-optimum, DQPP solutions. Firstly, we analyze how the mixture of ship sizes and time congestion impact the selected berth lengths. Secondly, a DQPP solution using the longest ship size as the length of all berths is compared against flexible berth size selection. Finally, we compare whether the 2in1 layout is better than the 1in1 layout. The analyses in this section provide practical indications for constructing good and avoiding bad QPP solutions.

3.6.1 Ship Size Mixing

In order to analyze how the mixture of ship sizes impacts the selected berth lengths, we designed the following computational experiment. The set of vessels V was divided into two subsets: set

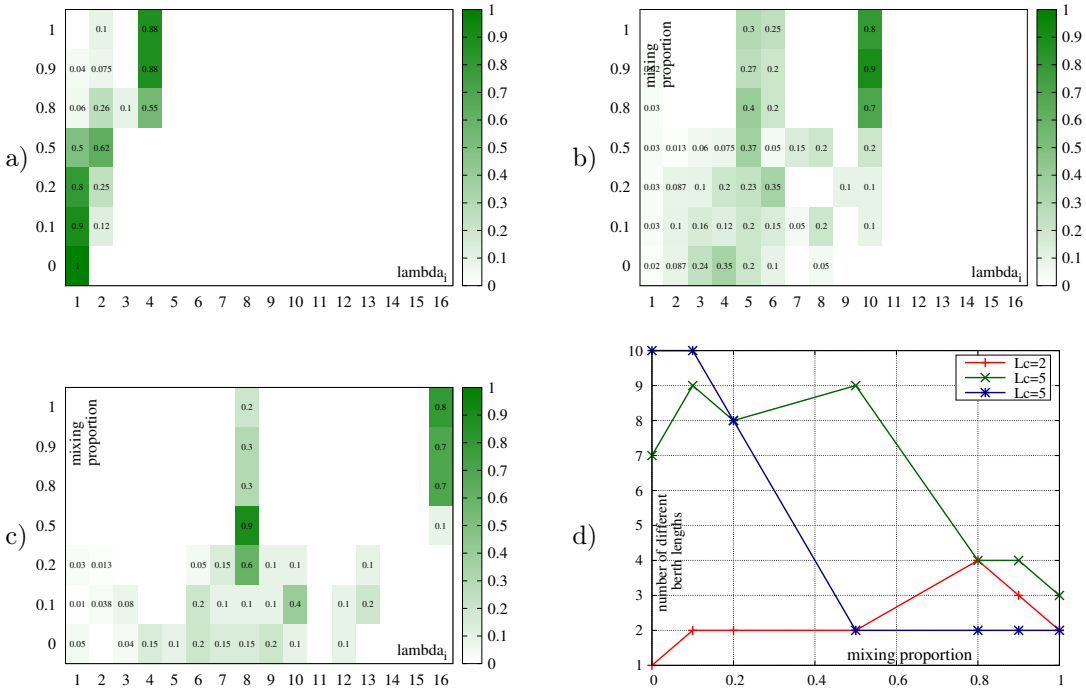


Figure 3.4: Berth lengths used in the solutions for 2in1. a) $L_{max} = 2$, b) $L_{max} = 5$, c) $L_{max} = 8$, d) Number of different berth lengths.

V_L comprising ships of fixed length L_{max}, and set V_S comprising ships with lengths generated from $U[1, L_{max}-1]$. The fraction of ships in sets V_L and V_S was controlled by a mixing proportion assuming values from 0 (ships from set V_S only) to 1 (only set V_L). Set B of admissible berth lengths included integer berth lengths up to $2L_{max}$, i.e. $B = \{1, \ldots, 2 * L_{max}\}$. This choice was dictated by the fact that 2in1 layout can take advantage of berth lengths of at most $2L_{max}$. Quay length was set to $Q = 16$ and number of ships to $n = 10$. The remaining parameters were generated as for the synthetic instances presented in Section 3.5. Ten test instances were generated for each presented data point. The instances are available from [94]. In Fig.3.3a-c and Fig.3.4a-c a fraction of the maximum possible berth number is shown as a heat map for berth lengths in range $1, \ldots, 16$ (horizontal axis) vs changing mixing proportion (vertical axis). The maximum possible berth number is $\rho_i = \lfloor Q/\lambda_i \rfloor$, for a berth of length λ_i. Fig.3.3 and Fig.3.4 show results for the same set of instances. In both sets of figures it can be noticed that

for a diverse set of ship lengths, also a diverse set of berth lengths is chosen as the solution. When long ship sizes dominate, it tends to be advantageous to use berth lengths of the longest ships size only. This may be advantageous for oceanic ports like Los Angeles and Long Beach where long container vessels abound [93]. The case of 2in1 layout is slightly different because when long ships dominate, berth lengths L_{max} are present in the solutions, but they are less frequent than length $2L_{max}$. Note that in Fig.3.3a and Fig.3.3b for mixing proportion equal 1 (i.e. when only ships of length L_{max} are present), berth lengths greater than L_{max} are selected. For example, in Fig.3.3a $L_{max} = 2$ but $\lambda_2 = 3, \lambda_2 = 4$ happen to be present in the solution. Though such sizes do not give any quality of service advantages, they are admissible and happen to be selected. Despite the fact that such solutions are valid, for simplicity of exposition, we will call them artifacts. Similar situation can be seen in Fig.3.4b for $L_{max} = 5$. Though $\lambda_2 = 6$ does not give advantage over berth lengths $\lambda_1 = 5, \lambda_3 = 10$ when all ships have length $L_{max} = 5$, yet $\lambda_2 = 6$ is admissible and happens to be selected. In Fig.3.3d and Fig.3.4d number of different berth lengths that happened to be present in the solutions are shown. Shape of the lines can be explained by two phenomena: firstly increasing L_{max} is increasing options for choosing different berth lengths, secondly when long ships tend to dominate in V, berth lengths equal to L_{max}, or $2L_{max}$, prevail.

Considering the above discussion, a simple managerial recommendation would be to use length L_{max} for 1in1 and $L_{max}, 2L_{max}$ for 2in1. Yet there are other factors determining DQPP solution which we study in the following sections.

3.6.2 Impact of Congestion

In this section we analyze impact of congestion in time on the berth choice. It is intuitively expected that with high traffic of ships of certain sizes also berth sizes should be chosen correspondingly and the longest berth size need not be the dominant solution. We will analyze the impact of congestion on berth size selection in two ways. Firstly, the correlation between sizes of chosen berths, instance congestion indicators, and mixing proportions will be evaluated. Secondly, the changes in the berth lengths used when all ships arrive together ($\forall_j, r_j = 0$), i.e. under congestion, will be studied. To this end, the test setting introduced in Section 3.6.1 will be exploited.

In Section 3.6.1 two sets of ships V_L and V_S were generated with lengths $\forall_{j \in V_L}, L_j = L_{max}$

and $\forall_{j\in V_S}, L_j \sim U[1, L_{max} - 1]$, where length L_{max} and the two sets mixing proportion were controlled experiment parameters. On the one hand, instance hardness indicator area ratio α, introduced in Section 3.5, will be used as a measure of the traffic congestion. On the other hand, the fraction of quay length used by berths shorter than L_{max}, that is $\phi = \sum_{\lambda_i<L_{max}} k_i\lambda_i/Q$, will be an indicator of the membership of short berths in the solutions. Results of this evaluation are shown in Fig.3.5a and Fig.3.5b for 1in1 and 2in1, respectively. Along horizontal axis area ratio α is shown. The higher α is, the more quay length is oversubscribed by ship area in time×space and the higher ship traffic congestion. Fractions of quay length occupied by berths shorter than L_{max} are shown along the vertical axis. It can be seen that with growing congestion (α increases) presence of short berth lengths (measured by ϕ) decreases.

Analogous relationship for the fraction of quay length Q in berths longer than or equal to L_{max}, i.e. for $\psi = \sum_{\lambda_i\geq L_{max}} k_i\lambda_i/Q$, is shown in Fig.3.5c and Fig.3.5d for 1in1 and 2in1, respectively. In this case the membership of long berths (ψ) increases with the congestion indicator (α). In both cases (ϕ vs α and ψ vs α) the correlations are quite apparent. The coefficients of correlation for various values of L_{max}, the experiment control parameter, are collected in Tab.3.5. Though not in all cases do the correlation coefficients exceed 0.7 absolute value, which is a rule of thumb threshold for strong correlation, the coefficients are quite high.

It could be concluded that short berths are indeed more present in the solutions when congestion indicator α is small. With the increase of congestion, longer berths are more frequently chosen. However, such conclusions would be superficial for at least two reasons: (1) congestion indicator α does not reveal ship size mixture, (2) there is a stronger determinant, than α, of berth size choices. Namely, mixing proportion between sets V_S and V_L of short and long ships. It can be verified in Tab.3.5 that correlations between the presence of short (ϕ) and long berths (ψ) is much stronger with the mixing proportion than with α. Moreover, area ratio α used in measuring congestion depends on the mixing proportion. These observations lead to extended conclusions: The presence of certain berth lengths (short or long) follow the strongest contributors to the congestion which are large vessels. Accommodation to the stream of the large ships prevail the needs of the shorter ships because large ships occupy a lot of resources (time and space) and in this way they dominate the short vessels. As a consequence, large ships traffic has priority in determining berth lengths. In the absence of large vessels, short ones may also incur congestion and in this case short berths are present in the solutions.

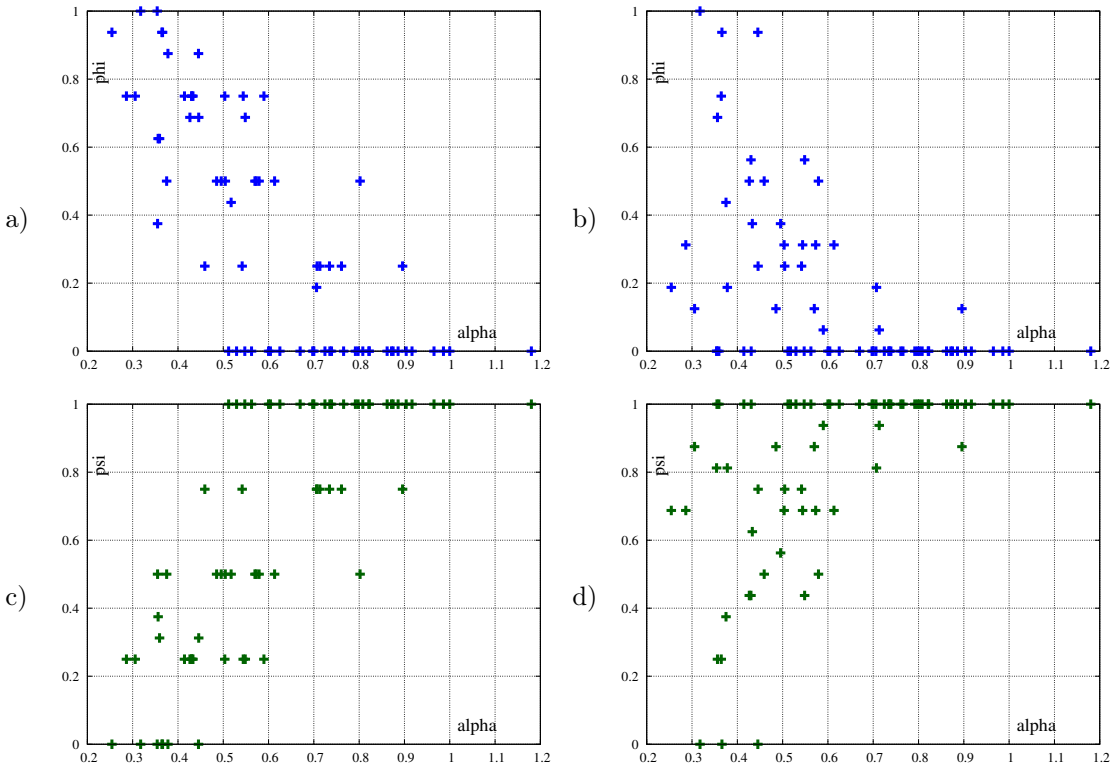


Figure 3.5: Fraction of berth length Q in short ($\lambda_i < L_{max}$) and long berths ($\lambda_i \geq L_{max}$) vs area ratio α for $L_{max} = 4$. a) short berths fraction ϕ vs α for 1in1, b) short berths fraction ϕ vs α for 2in1, c) long berths fraction ψ vs α for 1in1, d) long berths fraction ψ vs α for 2in1.

In the second experiment the setting from Section 3.6.1 was used again, however, with all ships arriving at the same time and in this way increasing time congestion. The same instances as in Section 3.6.1, yet with $\forall_j, r_j = 0$, were solved. The fraction of the maximum possible berth number was analyzed vs changing long V_L and short ships V_S mixing proportion. The difference between results obtained in Section 3.6.1 (Fig.3.3a-c, Fig.3.4a-c) and the current instance setting is shown in Fig.3.6. The change in the maximum possible berth number is shown as a heat map for certain berth lengths in range $1, \ldots, 16$ (horizontal axis) vs changing V_L, V_S mixing proportion (vertical axis). More frequent presence of certain berth length (than in Fig.3.3a-c, Fig.3.4a-c) is represented as a positive number (and color blue) and less frequent presence as a negative number (color red). Although there are some minor changes from the values shown in Fig.3.3a-c, Fig.3.4a-c, it is hard to identify strong change patterns. This is

Table 3.5: Correlations between presence of short ϕ and long berths ψ in quay length with area ratio α, and mixing proportion.

1in1							
L_{max}	2	3	4	5	6	7	8
correlation(α, ϕ)	-0.723	-0.669	-0.768	-0.722	-0.566	-0.664	-0.593
correlation(α, ψ)	0.777	0.683	0.779	0.720	0.505	0.568	0.573
correlation(mixing proportion,ϕ)	-0.992	-0.928	-0.942	-0.947	-0.886	-0.869	-0.869
correlation(mixing proportion,ψ)	0.963	0.940	0.935	0.941	0.873	0.850	0.869
2in1							
L_{max}	2	3	4	5	6	7	8
correlation(α, ϕ)	-0.727	-0.544	-0.545	-0.579	-0.457	-0.531	-0.410
correlation(α, ψ)	0.744	0.572	0.572	0.554	0.405	0.457	0.413
correlation(mixing proportion,ϕ)	-0.990	-0.739	-0.686	-0.783	-0.750	-0.653	-0.701
correlation(mixing proportion,ψ)	0.984	0.739	0.702	0.759	0.622	0.551	0.698
correlation(mixing proportion,α)	0.748	0.789	0.779	0.768	0.706	0.741	0.684

in line with the earlier observation that mixing proportion between long and short ships is a key determinant of selected berth length. Two phenomena can be observed: For small values of mixing proportion (short ship dominate) berth lengths shorter than L_{max} are slightly more present in the current solutions (Fig.3.6c,e,d,f in lower-left corner). This gives more space to schedule short ships. In the 2in1 case some berth lengths in range $[L_{max} + 1, 2L_{max}]$ are less frequent (Fig.3.6d,f middle-right of the pictures). These berths are longer than the longest ship and can be filled completely only by two ships sharing the berth. Using fewer, or different set of such berths is a sign of economizing on quay lengths. Moreover, it demonstrates combinatorial nature of the problem.

A managerial recommendations from this section can be summarized as follows: The chosen berth sizes follow the sizes of the ships causing the congestion. Since large vessels occupy a lot of time and space they prevail in berth size selection over short vessels. Only when large vessels are absent, can the short vessels dominate in berth size choice.

3.6.3 One Long Berth Length vs Flexible Berth Lengths

As a result of the discussion in Section 3.6.1, one may think that using berth lengths L_{max} for 1in1, or $2L_{max}$ for 2in1, can be an advantageous simplification in designing quay partition. In this section we verify analytically and experimentally three points of view on basing solutions on L_{max} or $2L_{max}$. In some sense we verify if indeed one "size fits all".

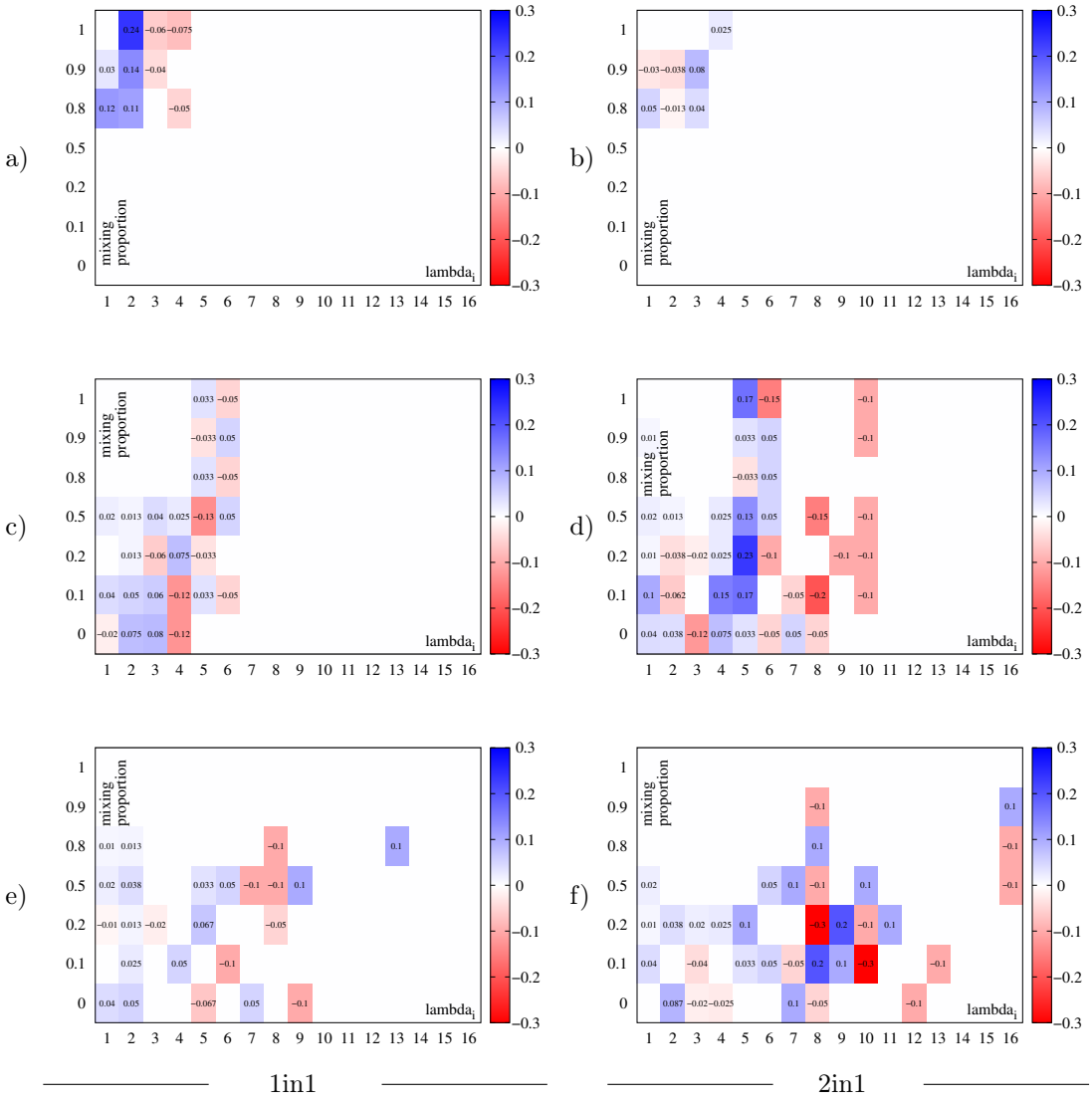


Figure 3.6: Change in the used berth lengths under congestion ($\forall_j, r_j = 0$). a) 1in1, $L_{max} = 2$, b) 2in1, $L_{max} = 2$, c) 1in1, $L_{max} = 5$, d) 2in1, $L_{max} = 5$, e) 1in1, $L_{max} = 8$, f) 2in1, $L_{max} = 8$.

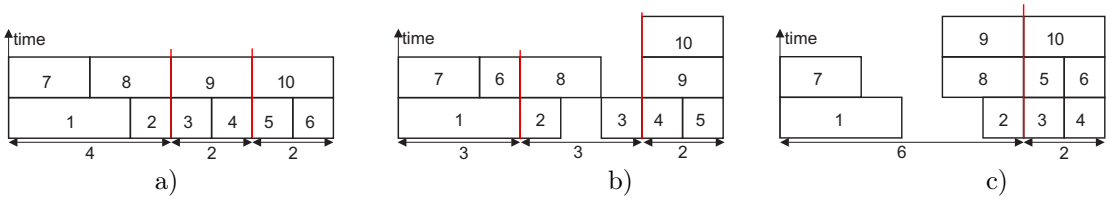


Figure 3.7: Observation 2. Schedules for a) $K_1 = (2, 0, 1, 0)$, b) $K_1 = (1, 2, 0, 0)$, c) $K_1 = (1, 0, 0, 1)$.

Observation 1. *A solution with $B = \{L_{max}\}$ may be $\Theta(n)$ times worse than a solution with flexible berth lengths.*

The $\Theta(n)$ notation [98] means that relative distance of solution with $B = \{L_{max}\}$ from the optimum solution is both lower- and upper-bounded by a linear function of ship number n. Thus, solutions with $B = \{L_{max}\}$ can be arbitrarily worse than the optimum. To verify the above observation consider the following example for 1in1. $B = \{Q/2\}, n = Q+1, r_j = 0, p_j = 1, L_j = 1, w_j = 1$ for $j = 1, \ldots, Q$, $r_n = 0, p_n = 1, L_n = Q/2, w_n = 1$ and Q is even. Since there is only one berth length equal to the half of the whole quay, two berths exist, and ships have to be served sequentially, resulting in $MWFT_1 = [2(\sum_{j=1}^{Q/2} j) + (Q/2+1)]/(Q+1) = (Q/2+1)^2/(Q+1)$. Conversely, consider the same set of vessels, but with $B = \{1, Q/2\}$. A partition with $Q/2$ berths of length 1 and 1 berth of length $Q/2$ is possible. Then in the first time unit $Q/2+1$ ships are served and in the second time unit $Q/2$ are served, resulting in $MWFT_2 = (3Q/2+1)/(Q+1)$. Hence, $MWFT_1/MWFT_2$ is $\Theta(n)$. A similar example can be constructed for 2in1. □

Thus, by Observation 1 *equipartitioning*, that is dividing a quay into equal length berths can be $\Theta(n)$ times worse than the optimum. The results in Fig.3.4 may suggest that using berth lengths $L_{max}, 2L_{max}$ in 2in1 is favorable. In the following observation we show that these two lengths may be absent from the optimum solution.

Observation 2. *The necessary berth lengths in 2in1 may be different than L_{max} and $2L_{max}$.*

Consider an example: $n = 10, \forall_j p_j = 1, w_j = 1$, $r_1, \ldots, r_6 = 0, r_7, \ldots, r_{10} = 1$, $L_1 = 3, L_2, \ldots, L_6 = 1, L_7, \ldots, L_{10} = 2$, $B = \{2, 3, 4, 6\}, Q = 8$. Thus $L_{max} = 3$ and berths of length $L_{max}, 2L_{max}$ are feasible. We will show that using $L_{max}, 2L_{max}$ results in inferior solutions. Suppose DQPP solution is $K_1 = (2, 0, 1, 0)$, that is there are two berths of length $\lambda_1 = 2$ and one berth of length $\lambda_3 = 4$. A solution with $F(K_1) = 1$ is shown in Fig.3.7a. A schedule for

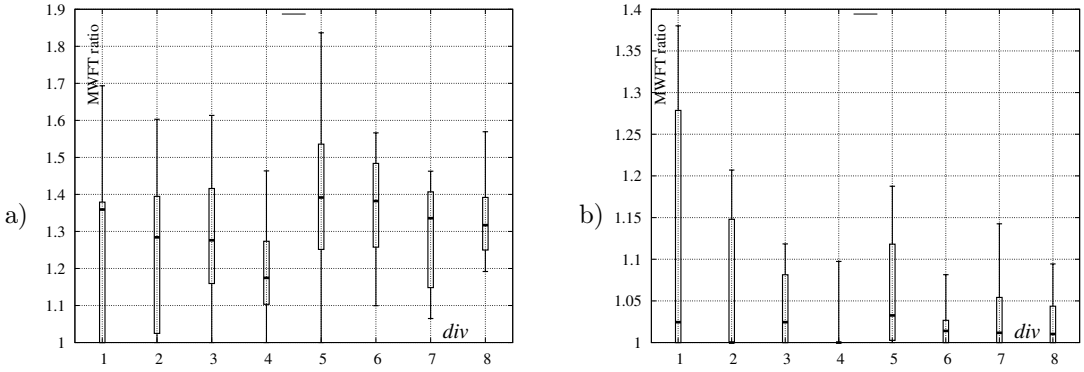


Figure 3.8: Ratio of MWFT vs number of allowed ship sizes *div*. a) 1in1, b) 2in1.

$K_2 = (1, 2, 0, 0)$, i.e. using two berths of $\lambda_2 = L_{max} = 3$, has $F(K_2) = 1.2$ and is shown in Fig.3.7b. A schedule for $K_3 = (1, 0, 0, 1)$, i.e. using a berth of $\lambda_4 = 2L_{max} = 6$, has $F(K_3) = 1.4$ and is shown in Fig.3.7c. Note that any partition with $\lambda_2 = L_{max} = 3$ prevents scheduling all ships with $r_j = 1$ in one time unit. A layout with $\lambda_4 = 6$ allows for only one vessel of length 3 and 3 vessels of length 1 in the first time slot. Thus, the optimal solution does not contain any berth of length L_{max} or $2L_{max}$.□

The above observations highlight combinatorial nature of DQPP. Therefore, we studied experimentally a less restricted setting. The experiment was designed as follows: Two sets of test instances were generated differing only in allowing or disallowing flexible berth lengths. Except for set B, the data were the same. In the first set of instances berth length was $B = \{8\}$, that is only berths of length 8 were allowed. In the second set of instances $B = \{1, \ldots, 8\}$, i.e. all integer berth sizes were possible. Vessel sizes were generated from $U[1, 8]$. However, the diversity *div* of lengths was controlled by limiting the number of different lengths. For example, for $div = 2$ all ships had one of two lengths each chosen from $U[1, 8]$. We set $n = 10, Q = 16$, the remaining vessel parameters were generated as described in Section 3.5. The instances are available from [94]. The test outcome is depicted in Fig.3.8 where quartiles of the ratio of the $MWFT$ for an instance from the first test set ($B = \{8\}$, restricted) to the $MWFT$ for the corresponding instance from the second set ($B = \{1, \ldots, 8\}$, flexible) are shown vs the diversity *div* of ship lengths. The worst case can be quite bad. Examples with the solutions worse by

84%, 38% were observed for 1in1 and 2in1, respectively. It can be seen that 2in1 and 1in1 perform apparently differently in this test because the median ratio is around 1.4 for 1in1 and 1.03 for 2in1. The latter is smaller because in 2in1 layout it is possible to pair ship lengths such that $L_i + L_j \leq \lambda_f = 8$ which is not possible in 1in1. Consequently, $MWFT$ for the solutions with $B = \{8\}$ are closer to the solutions when greater flexibility of berth length selection exists ($B = \{1, \ldots, 8\}$). No apparent tendency of the two ratios medians with changing diversity of ship sizes (div) can be seen. However, in 1in1 quartiles Q0 and Q1 tend to increase with growing div. With growing diversity div of ship sizes it is increasingly possible to build solutions where quay length is more flexibly distributed, more shorter berths are used, and $MWFT$ in the flexible case ($B = \{1, \ldots, 8\}$) can be smaller, which is not possible when $B = \{8\}$. Consequently the smallest ratios of the two $MWFTs$ increase. Similar phenomenon can be observed for 2in1 but for Q4, Q3. The greatest values of the ratios get milder with growing div because chances of more flexible pairing ship lengths in 2in1 layout are also increasing. Hence, the two values of $MWFT$ get closer to each other. Overall, the results of this section encourage using diverse berth sizes, and 2in1 scheme over 1in1 scheme, because the 2in1 is better in handling vessels on a set of limited berth lengths.

3.6.4 2in1 against 1in1

As already mentioned, 2in1 generalizes 1in1 because each solution for 1in1 is a feasible solution for 2in1. In the previous section it appeared that 2in1 offers better flexibility in constructing DQPP solution. A question arises, how much a solution using 2in1 layout can be better than a solution with 1in1 layout. Some light can be shed on this issue by comparing $MWFT$s of instances solved for 2in1 and 1in1 layouts. The results for such a direct comparison are collected in Fig.3.9 showing histogram of the 1in1 and 2in1 $MWFT$ ratios in a population of 2114 instances solved in the previous experiments. The first block represents the 907 instances where $MWFTs$ were equal. For over 50% of the cases the ratio was below 1.02 and for 90% of the cases it was below 1.16. The largest observed ratio of $MWFTs$ for the two types of layouts was $2821/1709 \approx 1.651$. Although the instances were generated to test specific phenomena described in the preceding sections, it can be concluded that in the verified set differences in $MWFTs$ are quite common, and on average are not large. The results in Fig.3.9 are in stark opposition to the results in Fig.3.8. It is possible because two different criteria in two different

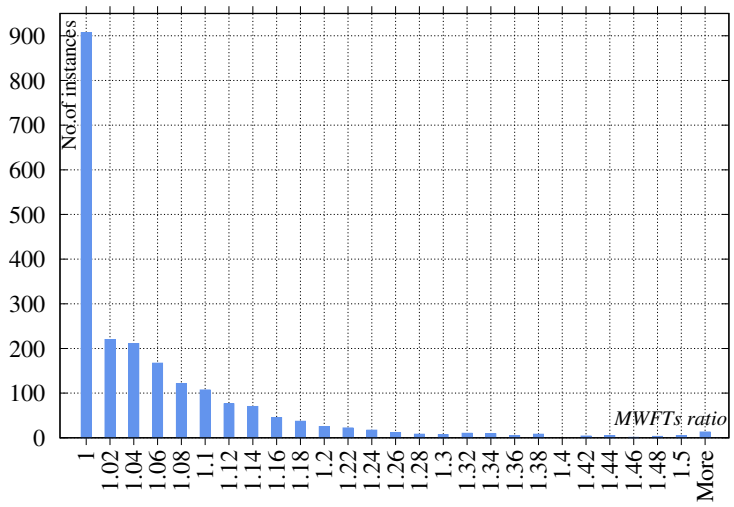


Figure 3.9: Distribution of 2in1 vs 1in1 objective ratios for the same instances.

experimental settings are evaluated. In Fig.3.8 it is shown that 1in1 is lacking the flexibility in handling diverse vessel sizes when berth size selection is limited (e.g. berths are already fixed). Contrarily, 2in1 is much better in dealing with such a situation. A direct comparison of $MWFT$s (Fig.3.9) does not reveal this feature of 1in1 because the selection of berth sizes was flexible both for 2in1 and 1in1.

3.7 Summary and Recommendations

In this chapter we introduced discrete quay partitioning problem in two variants: 2in1 and 1in1. Both variants were proven to be **NP**-hard even in very restrictive cases. Mixed integer linear programs were constructed for both cases. We evaluated time scalability of solving DQPP by the MIPs. It appears that this problem is hard to solve and only instances with moderate size could be solved to optimality, while larger instances can be dealt with when a MIP solver is used as a heuristic. Despite these obstacles we analyzed how ship size dispersion impacts selected berth lengths. We observed that for a diverse set of ship lengths a diverse set of berth lengths is also chosen to the solution. When the longest ships dominate then berths of length L_{max}

and $2L_{max}$ are the most frequently present in the optimum layouts. It can be also concluded that using the longest ship size can be an attractive heuristic solution to DQPP resulting in relatively limited loss of quality on average ($\approx$ 40% for 1in1 and $\approx$ 5% for 2in1). Yet, in the worst-cases such solutions may be bad. In case of traffic congestion quay partition follows sizes of the vessels with the greatest time-space demands. Next, we compared quality of the solutions in 2in1 and 1in1 setting and observed that typically quality differences are not large. Thus, the study conducted here gave hints on qualitative nature of DQPP solutions. The insights provided to terminal designers can be summarized as follows: (1) quay partitioning should depend on the future traffic, (2) use of 1in1 (discrete) layout is discouraged because it is not as flexible in dealing with diverse vessel size traffic as 2in1 (hybrid), (3) using all berths of the same size is suitable for the average case, (4) but the worst cases in such solutions can be very far from optimum, and hence, using all berths of the same size is not a good idea from the robustness point of view, (5) large ships occupy a lot of time and space, hence they prevail in berth size selection over short ships.

Chapter 4

Selecting Algorithms for Large Berth Allocation Problems

4.1 Algorithm Selection Problem for BAP

In this chapter algorithm selection for the berth allocation problem (BAP) under algorithm runtime limits is considered. For the purposes of strategic port capacity planning, BAP must be solved many times in extensive simulations, needed to account for ship arrival and service times uncertainties, and for alternative terminal designs. As argued in Section 2.3 specialized methods are necessary to solve large BAP instances appearing in the SQPP. Consider a motivating example: A stochastic STM is used to evaluate a quay partitioning and terminal layout. In order to extract quantitative features of one partition such as mean and dispersion of turnaround time, the partition is evaluated by solving BAP on 100 realizations of the probabilistic ship traffic model. Solving 10000-ship BAP even by a simple greedy algorithm can take as long as 4s (Table 4.4). To avoid possible bias of one greedy algorithm and improve solution quality we will run not one, but many algorithms. Suppose there are 50 alternative algorithms (Section 4.3) to be run. This will take at least $100 \times 50 \times 4s = 20000s$. Mind that we intend to evaluate not one possible berth layout, but hundreds and these layouts can be constructed by a probabilistic search algorithm. Furthermore, alternative ship traffic evolution scenarios may be examined. Each of these simulations needs to solve deterministic BAP as a subproblem, and

runtime allowance is the key design decision impacting statistical quality of the results. It is not possible to run any BAP algorithm blindly and respect the runtime allowance. Hence, there is a need for choosing algorithms solving many large BAP instances in a limited time. This, in turn, is the domain of the algorithm selection problem (ASP) [70].

ASP consists in developing methods for choosing best algorithms for the considered field of application. The criteria of algorithm performance are, e.g., runtime and solution quality. In the classic formulation, ASP is an instance classification problem, in which an instance is assigned to a class of instances solved in the best way by a certain algorithm. We depart from this classic approach and perceive ASP as a selection of a portfolio of algorithms which may be run in limited time. It can be expected that with increasing runtime limit the portfolio changes and the quality of the obtained solutions improves. The trade-off between solution quality and runtime limit, as well as the algorithm membership in the portfolio, will be studied on the example of BAP. In order to select the portfolio, a linear program minimizing the solution quality loss, subject to overall runtime limit is used. For the training and validating datasets, random instances and real ship traffic logs are used. The portfolio abilities to solve new instances are assessed.

Further organization of this chapter is the following. The next section presents the berth allocation problem. Section 4.3 is dedicated to the algorithms for BAP. Test instance datasets are introduced in Section 4.4 and the results of individual algorithm evaluation are given in Section 4.5. In Section 4.6 our approach to algorithm selection problem is outlined. Algorithm portfolios and their evolution in time are presented in Section 4.7. Section 4.8 is dedicated to evaluation of the portfolios. The last section recaps the results of this chapter. The main notations are summarized in Tab. 4.1. Only a small selection of the constructed algorithm portfolios and their performance evaluations are discussed in the chapter. A complete set of the results is provided in Appendix C.

4.2 Berth Allocation Problem Formulation

BAP considered in the following text is formulated as follows. A set of m berths of lengths λ_i, $i = 1, \ldots, m$, is given. There are n vessels defined by: arrival times r_j, lengths L_j, processing (service) times p_j, and importance w_j, $j = 1, \ldots, n$. Importance w_j of ship j represents the value of the ship for a terminal manager, e.g., the cost of mooring or the cost of the cargo. We

Table 4.1: Main notations in Chapter 4

	Berth Allocation Problem - parameters
λ_i	length of berth i
L_j	Length of ship j
m	number of berths
n	number of vessels
p_j	ship j processing time
r_j	ship j arrival time
w_j	value (or weight) of ship j
	Berth Allocation Problem - decision variables
c_j	the completion time of handling ship j;
$MWFT$	mean weighted flow time, the objective function
	Algorithm Selection Problem - parameters
$\mathcal{A}$	algorithm space, i.e., set of algorithms
$\mathcal{I}$	problem space, i.e., set of all instances
$\mathcal{I}'$	set of training instances
$q(a, I, T)$	the best quality score obtained by algorithm a for instance I by time T
$q_{\min}(I, T)$	the best quality score for instance I, obtained by any algorithm by time T
(t^j_{aI}, q^j_{aI})	jth pair of solution quality improvement to q^j_{aI} obtained at time t^j_{aI} by algorithm a for instance I
T	runtime limit
	Algorithm Selection Problem - decision variables
$Cost$	computational cost of a portfolio
x_a	1 if algorithm a is in the portfolio, 0 otherwise
G	regret, i.e., relative distance from $q_{\min}(I, T)$
u_{aI}	1 if algorithm a is instance I solver, 0 otherwise (in G minimization)

assume hybrid quay organization: each berth can accommodate at most two ships at the same time, provided the total length of two ships does not exceed the berth length (so called 2in1 berthing scheme). A solution of BAP is represented as a set of chains, where each chain is a sequence of ships mooring at a certain berth (see Fig. 4.1). Each berth has two chains, called the left and the right chains, representing the sequence of ships moored at the ends of the berth. The minimized objective is the mean weighted flow time $MWFT = \sum_{j=1}^{n}(c_j - r_j)w_j / \sum_{j=1}^{n} w_j$, where c_j is the completion time of handling ship j.

4.3 Algorithms

In this section we introduce the set $\mathcal{A}$ of algorithms solving BAP. We start with greedy algorithms and then proceed to more advanced methods.

4.3.1 Greedy Algorithms

Greedy algorithms are defined by two elements: the control structure and the sorting rule. The control structure determines the range of considered ships. The sorting rule sequences the ships.

Sorting rules

The following sorting rules were used. The ties were resolved arbitrarily.

First-Come, First-Served (FCFS): $r_1 \leq r_2 \leq \cdots \leq r_n$

Longest Ship First (LSF): $L_1 \geq L_2 \geq \cdots \geq L_n$

Shortest Ship First (SSF): $L_1 \leq L_2 \leq \cdots \leq L_n$

Longest Processing Time (LPT): $p_1 \geq p_2 \geq \cdots \geq p_n$

Shortest Processing Time (SPT): $p_1 \leq p_2 \leq \cdots \leq p_n$

Largest Area First (LAF): $p_1 L_1 \geq p_2 L_2 \geq \cdots \geq p_n L_n$

Smallest Area First (SAF): $p_1 L_1 \leq p_2 L_2 \leq \cdots \leq p_n L_n$

Weighted Shortest Processing Time (WSPT): $p_1/w_1 \leq p_2/w_2 \leq \cdots \leq p_n/w_n$

Greatest Importance (GI): $w_1 \geq w_2 \geq \cdots \geq w_n$

Greatest Importance – Shortest Processing Time (GISPT): this rule sorts the ships by importance first, like the GI rule, and then ships of the same importance are ordered as in the SPT rule.

Shortest Processing Time – Greatest Importance (SPTGI): this rule sorts the vessels by the SPT rule first, while ships with the same processing time are ordered by the GI rule.

Random (RND): this rule builds a random sequence of ships and is used as a reference algorithm to check whether other algorithms return useful solutions.

Control structures

Priority heuristics (Prio) proceed through the increasing ship ready times and ship service completion times. At each of these moments, the first ready ship is chosen from the list defined by the sorting rule. Then, the ship is assigned to the shortest available berth. If no ship fits any available berth, or if there are no more free berths at this moment, then we go to the next decision moment. The procedure is repeated until there are no ships to be scheduled.

List heuristics (List) rely strictly on the list of ships constructed by the sorting rule. As long as the list is not empty, the first ship is chosen and assigned to the earliest shortest berth, i.e., berths available at the earliest time are searched for first, and if there are more than one, the shortest is chosen. Berth availability time is the moment when the service of the last ship ends. The earlier unassigned intervals (if there are any) are not considered. If the chosen ship j is not ready, then the algorithm *waits* until its ready time r_j. Contrary to the previous control structure, the *list* structure allows a situation where a berth is not used by a ready ship, as the higher priority ship must be allocated first, although it arrives later.

k-Look-ahead heuristics (Lak) act like the priority structure, but at the decision points they take into account k future ship arrivals, and include the k future ships into the set from which the next ship to serve is chosen by the sorting rule. This control structure is considered for $k \in \{2, 5, 10\}$.

All the greedy algorithms can be implemented to run in $O(n \log n + m \log m + nm)$ time, where the first component accounts for the sorting rule, the second is the time of sorting berth lengths and the last is the time of finding berths for the ships. Note that for FCFS all control structures are equivalent because FCFS sorting rule coincides with the orders of all control structures.

We will use a shorthand notation to refer to the greedy algorithms concisely. A short name will consist of an abbreviation of the sorting rule name, followed by the abbreviation of the control structure name. For example, LPT-La2 is a heuristic with 2-look-ahead and LPT sorting.

Super Greedy algorithm (SG) is a combination of all greedy algorithms. It provides the best solution constructed by any greedy algorithm and its runtime is the sum of the runtimes of all the greedy methods.

4.3.2 Hill Climbers

Hill climber is a simple local search method. It starts from the best solution constructed by any greedy algorithm (cf. Section 4.3.1) and tries to improve the solution by moving ships from their current positions to different positions in the ship sequences (cf. Fig. 4.1). Hill climber is also used in post-optimization stage of GRASP algorithm (Section 4.3.3), so a hill climber can start from a solution built by the GRASP. The new ship positions must obey ship

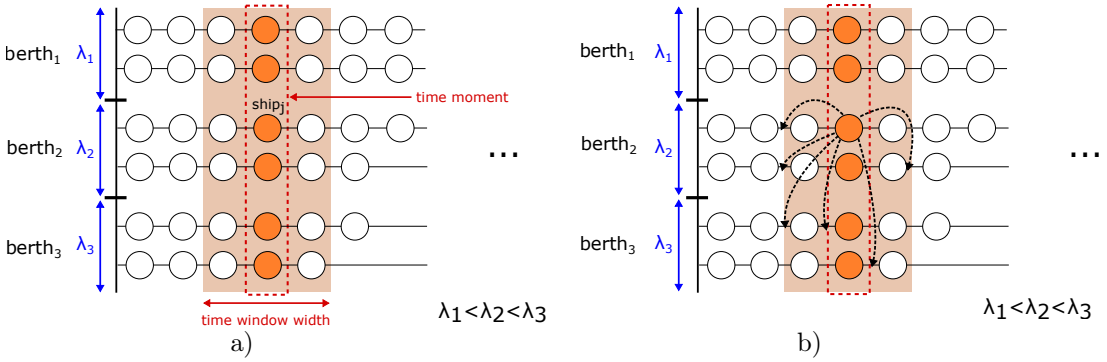


Figure 4.1: Solution neighborhood in HC and HC-A: a) the time window constraint, b) feasible moves in HC-A.

and berth length constraints. To ensure sufficient time to generate new solutions for large instances, the neighborhood size has been reduced by restricting the length WL of admissible move time window. Let st denote the current start time of handling the considered ship and let tt denote the start time of handling it in the potential target position. A move is possible only if $st - WL \leq tt \leq st + WL$ (cf. Fig. 4.1a). The objective function for the new solution is evaluated and if it improves, then the new solution is chosen as the current one. A limit NA on the number of attempts to execute a move from the current ship position has been also introduced. The hill climber works until there is no improving move or the time limit is exceeded.

We implemented three versions of the hill climber differing in the way of choosing ships to be moved. In the first variant, called HC, the search proceeds sequentially over all the ships. In the second variant, referred to as HC-A, a time moment τ is randomly generated in the existing schedule. All vessels whose interval of processing intersects with τ are tried for redistribution (cf. Fig. 4.1b). In the third variant, called HC-C, we select a time window of width SW on a random berth, starting at a random time position. All vessels in the time window on the selected berth are checked for relocation (cf. Fig 4.2a). In the three versions, control parameters have been independently tuned to $NA = 10, WL = 10$, and in HC-C to $SW = 10$ (see [92] for the tuning process details).

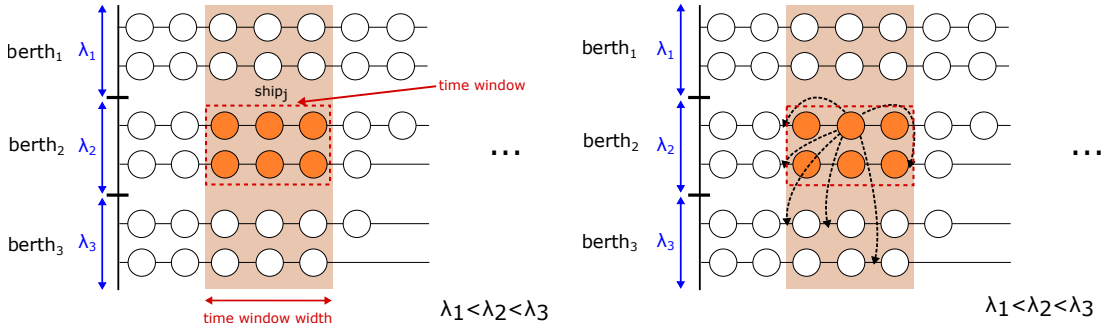


Figure 4.2: Solution neighborhood in HC-C: a) visualization of the constraints, b) feasible moves.

4.3.3 Greedy Randomized Adaptive Search Procedures

Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic which randomizes greedy algorithms [53, 73]. In our case GRASP proceeds in two steps. First an initial solution is built: A set of k vessels is chosen from the top of a list built according to some sorting rule. The k ships constitute the so-called restricted candidate list (RCL). Then, one vessel is drawn randomly (with equal probability) from the RCL and appended to the earliest shortest berth. Thus, berths are checked first in the order of increasing availability times and then in the order of lengths. The above procedure is repeated until all n ships are scheduled. This first part of the algorithm is repeated until time limit tl_1 elapses. The best solution is always retained. In the second part, a version of the hill climber (HC-A, see Section 4.3.2) is applied for post-optimization. The hill climber is run until the algorithm is suspended on the overall runtime limit. Six implementations of GRASP have been tested and tuned [92]: GRASP-LSF ($k = 20, tl_1 = 600$s, sorting rule LSF, post-optimization HC-A), GRASP-LAF ($k = 20, tl_1 = 600$s,LAF,HC-A), GRASP-GI ($k = 10, tl_1 = 600$s,GI,HC-A), GRASP-GISPT ($k = 20, tl_1 = 600$s,GISPT,HC-A), GRASP-SPTGI ($k = 10, tl_1 = 600$s,SPTGI,HC-A), GRASP-3600 ($k = 10, tl_1 = \infty$,GISPT,no post-optimization).

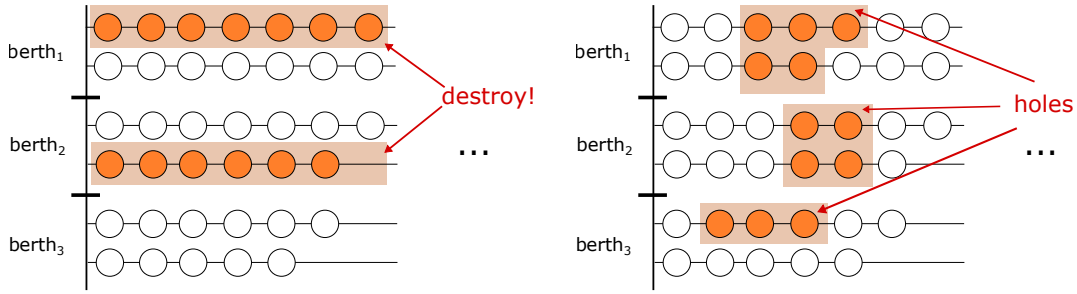
4.3.4 Iterated Local Search

Iterated Local Search (ILS) starts from the best solution S returned by any greedy algorithm, and then iteratively destroys and reconstructs parts of S. ILS stops on reaching a time limit.

In the first variant, called ILS-A, whole chains in the current solution are dismantled (Fig.

4.3a). Let $\varepsilon \in [0,1]$ be the fraction of the whole number of chains which should be destroyed. The chains are chosen randomly with equal probability. A constraint has been imposed that in each iteration ILS-A has to dismantle at least two chains from at least two different berths. The ships from the dismantled chains are sequenced by a greedy algorithm and appended in this sequence to the chains being reconstructed. Suppose two chains exist on some berth: a new one that is being reconstructed, and an old one which is an unmodified chain remaining from the initial solution S. For some ship j, a berth i of length $\lambda_i \geq L_j$ available at the earliest time greater than or equal to r_j, is searched for first. If more than one berth meets the above conditions, then the shortest one is chosen. The calculation of berth i availability time bat_i takes into account two cases. Let τ be the time when the service of the last ship in the new chain ends. If no ship is scheduled in the old chain at time τ, then $bat_i = \tau$. Otherwise, bat_i is the time when the service of this ship in the old chain ends. If both chains are new, then the maximum of their completion times is used as bat_i. Assigning ship j to a berth may cause a conflict with the schedule of the old chain. Namely, the newly inserted ship j and some ship k scheduled in the old chain in the interval $[bat_i, bat_i + p_j]$ may have lengths such that $L_j + L_k > \lambda_i$. If it is the case, then the ships in the left chain have priority, i.e. the ship at the left end of the berth is scheduled first, while the ship in the right chain is scheduled as the second. The ships remaining in the tail of the old (unmodified) chain are delayed accordingly. All greedy algorithms are applied to reconstruct the solution and the best solution is selected. We implemented two versions of this method. In the first one, referred to as ILS-A, ε was tuned to $\varepsilon = 0.3$. The second version, referred to as ILS-A0, always destructs and reconstructs two chains.

In the second variant, referred to as ILS-C, ships are removed from the current solution to create „holes" in the existing schedule (see Fig. 4.3b). It is assumed that holes are created in the chains representing the schedules on the berths, and there can be at most one hole in a chain. Let $\varepsilon \in [0,1]$ be the fraction of the solution that must be destroyed, i.e. $n\varepsilon$ ships are removed from the current solution. The ship removal process progresses in three stages. Firstly, chains are selected with equal probability until collecting chains comprising at least $n\varepsilon$ ships. Secondly, $n\varepsilon$ ships are distributed randomly between the selected chains with uniform probability. Let z_i be the number of ships to be removed from chain i. Then z_i consecutive ships are removed from chain i starting at a position chosen with uniform probability along the chain length. In the reconstruction process the ships are ordered by some greedy algorithm and reinserted into



a) b)

Figure 4.3: Operation of Iterated Local Search variants: a) ILS-A, b) ILS-C.

holes in the schedule. When some ship j is chosen to be scheduled, it is assigned to a hole in the earliest shortest feasible berth. The availability time of a position in a hole is determined in the following way. Suppose k is the index of the last ship in the chain before a hole. Indices $k+1, \ldots, k+a$ represent ships already inserted in the hole. Availability times of the positions in the hole are times $c_k, c_{k+1}, \ldots, c_{k+a}$ when the services of ships $k, k+1, \ldots, k+a$ end. If some position k' in a hole of chain i is selected, then the ships in the positions later than k' are delayed to allow inserting the considered vessel j. If there is a conflict with the ships in the other chain of the berth, then they are also delayed. Of the two conflicting chains, the left one is always given preference. All greedy algorithms are applied in sequencing the ships to be reinserted into the schedule and the best obtained solution is selected. For ILS-C, the control parameter ε has been tuned to 0.1 (see [92] for the tuning details).

4.4 Test Datasets

We used five datasets for the testing purposes. Four datasets were randomly generated, and the fifth one represented real ship traffic. Unless stated to be otherwise, in the random datasets parameter values were drawn as follows: $n \sim U[1,1000]$, $m \sim U[1,100]$, $r_j \sim U[0,1000]$, $p_j \sim U[1,24]$, $w_j \sim U[1,1000]$, $L_j, \lambda_i \sim U\{200,215,290,305,400\}$. By $\sim U[a,b]$ we denote that a certain parameter is generated from a discrete uniform distribution with integer values in the range $[a,b]$. The notation $\sim U\{x,\ldots,y\}$ means that the parameter values are chosen with a discrete uniform distribution from the set $\{x,\ldots,y\}$. The set of ship lengths L_j represents popular

container ship classes like Suezmax and (New) Panamax. Each configuration of the tests comprised a population of 100 instances. Two series of random instances were used to test the impact of parameters n and m on the algorithm performance. In such tests, a range of the tested parameter was traversed by explicitly setting values of the tested parameter, while the remaining parameters were randomized. For example, in the tests of the impact of the number of ships n, the following set of n values was considered: $\{2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000\}$. This means that 100 instances were generated with $n = 2$ while the remaining parameters were randomly generated as described above. The 100 instances with $n = 2$ were solved by the tested algorithms to evaluate their performance. Next, values $n = 5, 10, \ldots, 10000$ were examined in the similar manner. This set of instances will be referred to as *random instances N*. In the examination of the impact of m, the tested values were $m \in \{1, 2, 5, 10, 20, 50, 100\}$ while other parameters were generated as described above. These instances will be referred to as *random instances M*. Overall, we used random datasets: (1) all 1200 random instances N, (2) 100 random instances N with $n = 10000$, (3) all 700 random instances M, (4) 100 random instances M with $m = 2$.

In the fifth dataset, referred to as *real* instances, authentic data on ship arrival times (r_j), processing times (p_j), and lengths L_j, were obtained from the Automatic Identification System (AIS), a global radio vessel tracking system. In the real instances, ship traffic of 8 ports in 2016 is represented. The ports with the number of ships were: Gdańsk – 465 ships, Long Beach – 995, Los Angeles – 1310, Le Havre – 2277, Hamburg – 3273, Rotterdam – 3997, Shanghai – 11197, and Singapore – 18413. Ship sizes range from 75m (Shanghai) or 87m (Singapore) to 400m. Berth lengths were obtained from the publicly available port authority data. Their length range is [200m,855m] with 500m median. The datasets are available at [60].

4.5 Initial Evaluation of Algorithms

In this section we report on the initial evaluation of the individual algorithms performance on the random datasets. The goal was to choose "the best" algorithm for the port simulation. The algorithms were implemented in C++ and compiled with `g++` ver. 4.8.2. All computational experiments were conducted in 2019 at Poznań Supercomputing and Networking Center on `Eagle`/`Altair` cluster, which at that time had Intel Xeon E5-2697 CPU@2.6 GHz, at least

64GB RAM per node, and Scientific Linux CERN 6.7. All the algorithms were sequential and had a one hour runtime limit.

When solving a set of instances, a population of results is obtained. Various algorithms have various result quality distributions. Given the result quality distribution, it must be decided which algorithm is the best. It is not possible to reduce the distributions to a single numerical score without losing some information. Depending on the choice of the distribution performance metrics (e.g. median, average), different algorithms can be chosen as the best [16]. Nevertheless, some pragmatic approach is needed to compare solutions provided by various algorithms. In this study we will use the number of wins, i.e., the number of instances for which some algorithm provided the best solution. Similarly, the number of unique wins is the number of instances for which some algorithm provided the best solution uniquely. An advantage of these metrics is that they are unambiguously defined even if the algorithm provides no solution (it does not win). A disadvantage of the number of (unique) wins is that an algorithm which is always, e.g., the second-best, is outperformed by an algorithm which is always the worst except for one instance when it gives the best solution. Though this situation is perfectly in line with the idea of algorithm portfolios (Section 4.6), we will use classic population statistics (median, average) as secondary quality metrics.

Table 4.2: Performance in the experiments with changing n (1200 instances).

#	Wins (p-value)		Relative distance (No. of σ s)		Unique wins	
1	ILS-A	728(0)	ILS-A0	1.0367(13.8)	ILS-C	217
2	ILS-A0	727(0)	ILS-A	1.0371(14.7)	ILS-A	196
3	ILS-C	725(0)	ILS-C	1.104(16.1)	ILS-A0	194
4	HC-C	514(0.010)	HC-C	1.107(18.7)	HC-C	32
5	HC	494(0.067)	HC	1.110(18.9)	GRASP-3600	10
6	HC-A	484(0.139)	HC-A	1.139(20.9)	HC	5
7	GRASP-3600	479(0.190)	SG	1.182(23.5)	HC-A	4
8	SG	479(0.190)	GRASP-3600	1.200(20.3)	GRASP-GISPT	2
9	GRASP-GISPT	471(0.293)	GRASP-GI	1.210(20.6)	GRASP-SPTGI	1
10	GRASP-SPTGI	470(0.307)	GRASP-GISPT	1.216(19.9)		
11	GRASP-LAF	469(0.322)	GRASP-SPTGI	1.219(20.2)		
12	GRASP-LSF	469(0.322)	GI-Prio	1.222(22.9)		
13	GRASP-GI	469(0.322)	GISPT-Prio	1.223(22.9)		
14	SAF-Prio	469(0.322)	SPTGI-Prio	1.269(20.7)		
15	SPTGI-Prio	468(0.337)	SAF-Prio	1.287(25.8)		
16	SPT-Prio	468(0.337)	SPT-Prio	1.294(20.9)		
17	GI-Prio	468(0.337)	RND-Prio	1.423(28.1)		
18	GISPT-Prio	468(0.337)	GRASP-LSF	1.555(27.1)		
19	LPT-Prio	462(0.433)	GRASP-LAF	1.564(29.1)		
20	LAF-Prio	461(0.450)	SSF-Prio	1.585(27.5)		
21	SSF-Prio	461(0.450)	WSPT-Prio	1.783(24.8)		
22	RND-Prio	460(0.467)	FCFS-List	1.816(24.6)		
23	LSF-Prio	459(0.483)	FCFS-Prio	1.816(24.6)		
24	WSPT-Prio	459(0.483)	FCFS-La10	1.816(24.6)		
25	FCFS-List	458(0.500)	FCFS-La2	1.816(24.6)		
26	FCFS-Prio	458(0.500)	FCFS-La5	1.816(24.6)		
27	FCFS-La10	458(0.500)	GI-La2	2.102(7.6)		
28	FCFS-La2	458(0.500)	GISPT-La2	2.103(7.6)		
29	FCFS-La5	458(0.500)	LPT-Prio	2.287(23.8)		
30	WSPT-La2	370(0)	WSPT-La2	2.322(12.3)		
31	WSPT-La5	260(0)	SAF-La2	2.497(11.7)		
32	WSPT-La10	165(0)	SPTGI-La2	2.505(11.5)		
33	LSF-La2	99(0)	SPT-La2	2.511(11.5)		
34	SSF-La2	93(0)	SSF-La2	2.669(14.8)		
35	SPT-La2	92(0)	RND-La2	2.692(14.3)		
36	SPTGI-La2	88(0)	LAF-Prio	2.969(23.1)		
37	LPT-La2	87(0)	LSF-Prio	3.099(18.6)		
38	GI-La2	86(0)	LPT-La2	3.956(11.6)		
39	GISPT-La2	86(0)	WSPT-La5	3.967(12.8)		
40	SAF-La2	82(0)	GI-La5	4.282(16.3)		

Solution quality for large runtime limit. Let us start with the comparison of algorithm performance for 1200 random instances N (dataset 1) and one hour runtime limit shown in Tab.4.2. The algorithms are ranked according to the number of wins, relative distance from the best solution found and the number of unique wins. The first 40 positions of the comparison are shown. The left panel of Tab.4.2 features the number of wins with p-values. The p-values were obtained by comparing two population proportions, where the reference population was the set of results of the FCFS-Prio algorithm. A p-value is the probability of the two populations of wins (of the given algorithm and of the FCFS-Prio) being the same. The central panel of Tab.4.2 gives the average relative distance and the number of σs from the best solution found. The relative distance from the best solution is the ratio of the $MWFT$ obtained for a given instance and the $MWFT$ of the best solution found. The number of σs is the average distance from the best solution found, divided by the standard deviation of the algorithm distances from the best solutions. This test verifies whether the two distributions of fractions are the same. This statistic is used because the corresponding p-values were too close to 0, which did not allow us to assess the algorithm results. The p-values and σ scores verify statistical soundness of the results. Low p-values and high σs mean that the populations of solutions are indeed different. The third panel in Tab.4.2 provides the number of unique wins.

Tab.4.2 shows that from position 30 the number of wins collapses. The positions with small number of wins are occupied by the greedy algorithms with *list* and *look-ahead* control structures. The *list* control structure is inefficient because it waits for specific vessels chosen by some sorting rule while ignoring the ready ones. The *look-ahead* control structure suffers from a similar deficiency: future ship arrivals delay scheduling the currently ready ships by giving preference to the future ships elected by the sorting rule. The best greedy heuristics with the *list* or *look-ahead* control structure use FCFS sorting rule, which is the order of the ships arrivals itself, thus breaking the spell of waiting for the future ships. This situation requires a comment. It is a common practice to reserve certain berths for the biggest ships and the berths indeed wait for those ships. However, in the *list* and *look-ahead* heuristics the sorting rules may completely disconnect the order of serving the ships from their proximity in time. Algorithm SAF-Prio tops the list of greedy heuristics, but the difference in the number of wins between the heuristics with priority control structure is roughly only 2%. Also, the p-values do not support the claim that priority heuristic results are significantly different. Thus, with respect to the number of

Table 4.3: Heuristic performance for different runtime limits (1200 instances).

Top 3 heuristics with the highest number of wins						
time limit	1st		2nd		3rd	
100ms	SG:	817	GISPT-Prio:	613	GI-Prio:	580
1s	SG:	759	ILS-C:	581	ILS-A0:	580
10s	SG:	732	ILS-A0:	623	SPTGI-Prio:	623
100s	ILS-A0	684	ILS-C:	664	ILS-A:	651
1000s	ILS-A0	715	ILS-A:	708	ILS-C:	698
Top 3 heuristics with the highest number of unique wins						
time limit	1st		2nd		3rd	
100ms	GISPT-Prio:	75	SAF-Prio:	44	GI-Prio:	42
1s	SPTGI-Prio:	75	ILS-C:	63	GISPT-Prio:	63
10s	SPTGI-Prio:	155	ILS-C:	105	ILS-A0:	80
100s	ILS-C:	157	ILS-A0:	146	ILS-A:	113
1000s	ILS-C:	191	ILS-A0:	179	ILS-A:	172
Top 3 heuristics with the minimum distance from the best found						
time limit	1st		2nd		3rd	
100s	SG:	1.122	GI-Prio:	1.158	GISPT-Prio:	1.158
1000s	SG:	1.170	GRASP-3600:	1.192	GRASP-GI:	1.197

wins, *priority-based* greedy heuristics perform similarly. The same conclusion can be drawn for the GRASP and SG methods. Only for methods ILS-A, ILS-A0, ILS-C and HC-C the claim that their solutions are better than FCFS's is supported by the p-values (because probability of error in such a claim is below 0.01). The central panel of Tab.4.2 shows that from position 27 the average distance from the best solution is over 100%. The big number of σs supports the claim that the algorithm solution populations are indeed different from the best solutions found. The third panel shows a ranking of the algorithms providing unique best solutions. It is a natural recommendation of the methods which should be included in the algorithm portfolio, provided runtime allowance is sufficient. On the one hand, Tab.4.2 allows us to identify the best algorithms because the top positions of the three rankings in Tab.4.2 coincide. On the other hand, it ignores the time cost of obtaining the solutions. This limitation is addressed in the rest of this chapter. Let us also note that *list* or *look-ahead* control structures performed poorly here. However, in Section 4.7 we will show that at least some of the algorithms using them can be useful in solving certain instances within some runtime limits.

Solution quality-runtime trade-off. Let us analyze how the algorithm ability to cover a set of instances with the best solutions changes with the runtime. Tab.4.3 lists the best three

algorithms in the sense of (unique) wins and minimum distance from the best solution found within certain runtime limit. Random instances N (dataset 1, 1200 instances) were used to build Tab.4.3. For random instances M, the conclusions were similar so we do not report them here. Different runtime limits of 0.1s, 1s, 10s, 100s, 1000s represent various levels of acceptable runtime to solve long-time horizon BAPs. It can be seen that for short runtime limits the greedy algorithms dominate in the number of wins. The number of unique wins of individual greedy algorithms is not big. Combining them into the composite SG algorithm is profitable because SG takes the first position in the number of wins up to the 10s runtime limit. However, since SG is a combination of heuristics, it cannot win uniquely and it is absent in the central part of Tab.4.3. With the increasing runtime limit, metaheuristics come forward and ILS-type methods dominate with respect to the number of (unique) wins. In the last part of Tab.4.3, runtime limits shorter than 100s are omitted because there are instances (e.g. with $n = 10000$) which were not solved by any algorithm in time shorter than 13.9s. The average distance in the population of 1200 random instances N is not defined for such runtime limits. For runtime limits 100s and 1000s, SG, GI-Prio, GISPT-Prio and even GRASP-type methods are effective. It can be concluded that under short runtime limits greedy algorithms are very useful, but no single greedy method is universally best even for short runtimes. More complex algorithms, such as metaheuristics, should be applied for longer runtime limits.

The criteria of the number of (unique) wins represent algorithm ability to cover sets of instances with the best solutions, but do not reveal solution quality when the solutions are not the best. Therefore, in Tab.4.4 we show algorithms nondominated in the runtime-quality trade-off. The quality indicator is the median relative distance of the $MWFT$ from the best known solution in a population of 100 random instances N with sizes: $n = 100, 1000$ and 10000. The median time of delivering the best solution constructed by a certain algorithm is given (not the total runtime, which is one hour in the case of metaheuristics). Notice that for the shortest runtimes greedy algorithms with the sorting rules based on ship processing times and weights dominate. Then, GRASP-like, HC-like and finally ILS-type methods follow. In such an evaluation setting, only 9-10 algorithms are nondominated for some fixed n and 19 unique heuristics are present overall in Tab.4.4.

Let us summarize the above evaluation. Some choice of the algorithms can be done on the basis of Tables 4.2, 4.3 and 4.4. However, a number of difficulties should be recognized: (1) The

Table 4.4: Nondominated heuristics in the $MWFT$ vs runtime comparison (medians).

$n = 100$			$n = 1000$			$n = 10000$		
name	time	quality	name	time	quality	name	time	quality
FCFS-List	0.487ms	1.0571	SPT-La10	56.66ms	2.1610	SPT-List	4569ms	1.3279
WSPT-La10	1.206ms	1.0545	SPT-La5	58.32ms	1.9439	SPT-Prio	4668ms	1.2221
WSPT-La5	1.317ms	1.0536	SPT-Prio	62.75ms	1.7519	SPT-La10	4734ms	1.2166
SSF-Prio	2.029ms	1.0261	SPTGI-Prio	68.04ms	1.7142	SPTGI-List	4879ms	1.1934
GRASP-3600	2.376ms	1.0252	GI-Prio	89.24ms	1.5870	GRASP-SPTGI	439.4s	1.1077
HC-A	90.45ms	1.0201	HC-A	5537ms	1.4105	GRASP-GISPT	449.1s	1.0383
HC-C	91.48ms	1.0189	HC-C	6956ms	1.3432	HC-A	649.4s	1.0133
ILS-A	326.1ms	1.0039	ILS-A	1279s	1.0122	HC	959.7s	1.0106
ILS-A0	389.8ms	1.0029	ILS-A0	1311s	1.0006	ILS-A	1529s	1.0065
ILS-C	154.6s	1						

selection depends on the ranking method. (2) Though some algorithms top a number of the rankings, no algorithm is uniquely best. (3) The choice depends on the runtime allowance. (4) In some rankings, Super Greedy is quite effective and it is a good idea to have such an algorithm. Observations (1) and (2) inspire further reflections. In order to remain efficient in many ranking methodologies, it seems reasonable to use a set of algorithms instead of "the single best one". Observation (3) encourages us to use greedy algorithms for speed and metaheuristics for quality. Yet, since metaheuristics need time to run, how to swap smoothly the greedy algorithms for the metaheuristics? The impact of runtime allowance is related to the instance sizes. Furthermore, a more general remark can be made: instead of classically asking what the time to get the optimum solution is, one should rather ask what the best solution that can be got in the given time is. Regarding remark (4), Super Greedy runs all greedy algorithms blindly, even those found to be quite ineffective. Some of the component algorithms of SG may be eliminated without loss of solution quality. Thus, a method excluding some SG components is necessary.

Overall, it can be concluded that the choice of algorithms for solving BAP is quite a confusing task, and the single "best algorithm" approach is not satisfactory. A more thorough approach is necessary to choose the algorithms for BAP. This is the subject of the algorithm selection problem and the subsequent sections.

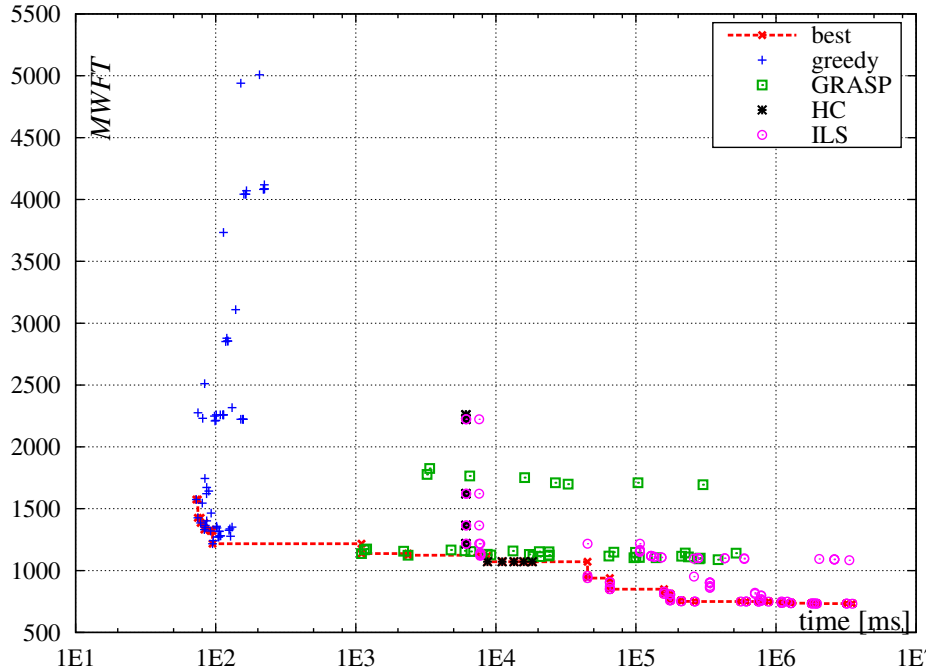


Figure 4.4: Quality and runtime of all solutions for an exemplary instance with $n = 1000$ and $m = 5$. A curve of best solutions found in limited time $q_{\min}(I, T)$ is shown. Algorithms of the same type are shown under the same label.

4.6 Algorithm Portfolio Selection Methods

Consider now algorithm selection for BAP. Though BAP is not a new problem, the features of BAP instances are not as well studied as for some classic **NP**-hard problems [78]. Hence, using instance features in the algorithm selection for BAP requires an independent study. Here we will formulate two new methods of algorithm selection using algorithm portfolios. These methods differ from the classic ASP approaches in: (1) departing from instance classification, (2) constructing a mapping from a set of instances to a portfolio of algorithms, (3) changing the portfolio with runtime limit T, (4) using the best solution found by T as solution quality reference. Thus, runtime-quality trade-off is the crux of the two methods.

Assume that a set $\mathcal{I}'$ of training instances and a set $\mathcal{A}$ of algorithms are given. For each pair $(a, I) : a \in \mathcal{A}, I \in \mathcal{I}'$ a sequence of pairs $(t^1_{aI}, q^1_{aI}), (t^2_{aI}, q^2_{aI}), \ldots, (t^{K_{aI}}_{aI}, q^{K_{aI}}_{aI})$ representing solutions built by a for I over time is given. The t^j_{aI} are execution times and q^j_{aI} are solution $MWFT$ values. The pairs satisfy conditions $t^1_{aI} < t^2_{aI} \cdots < t^{K_{aI}}_{aI}$ and $q^1_{aI} > q^2_{aI} > \cdots > q^{K_{aI}}_{aI}$

(see Fig.4.4). If some algorithm a does not improve solution quality over time, then only one pair (t^1_{aI}, q^1_{aI}) is given. An important decision is defining runtime $t(a, I, T)$ of algorithm a on instance I with the runtime limit T, especially if a does not stop by T or it does not produce any solution by T. The decision is related to the practical control over the algorithm runtimes when applied to a new (previously unseen) instance. We assume the simplest solution: each algorithm in the portfolio is allowed to run for at most T units of time. If algorithm a stops at $t^1_{aI} \leq T$, then we accept $t(a, I, T) = t^1_{aI}$. Otherwise, a will be suspended at T and we assume $t(a, I, T) = T$. The former case applies, e.g., to one-pass methods like greedy algorithms. The latter case applies both to the slow one-pass methods and to metaheuristics which can run endlessly. Thus, T is a timeout value. Let $q_{\min}(I, T) = \min_{a \in \mathcal{A}}\{q^j_{aI} : t^j_{aI} \leq T\}$. In other words, $q_{\min}(I, T)$ is the value of the best solution for instance I, obtained by any algorithm by time T (cf. Fig.4.4). If no solution is constructed by any algorithm by time T, then $q_{\min}(I, T)$ is undefined. Consequently, there can be values of T for which some instances have no solutions. We assume that for the considered values of T, the longest shortest runtime for any instance $\max_{I \in \mathcal{I}'} \min_{a \in \mathcal{A}}\{t^1_{aI}\} \leq T$ is satisfied, i.e. for each instance I there is at least one algorithm a such that $t^1_{aI} \leq T$. Let $y(a, I, T) = 1$ if $\exists j : t^j_{aI} \leq T, q^j_{aI} = q_{\min}(I, T)$, otherwise $y(a, I, T) = 0$. That is, $y(a, I, T)$ indicates if algorithm a is capable of providing for instance I the best solution known by time T. Note that some algorithm may have both $t(a, I, T) = T$ and $y(a, I, T) = 1$ when it finds the solution with quality $q_{\min}(I, T)$ before T, but continues to run.

Let $x_a \in \{0, 1\}$ be a binary variable equal to 1 if algorithm $a \in \mathcal{A}$ is included in the portfolio, 0 otherwise. The algorithm portfolio can be selected using the following integer linear program (ILP). The portfolio will be referred to as COVER PORTFOLIO (CP for short).

$$CP: \quad \min Cost \tag{4.1}$$

$$\sum_{a \in \mathcal{A}} y(a, I, T) x_a \geq 1 \quad \forall I \in \mathcal{I}' \tag{4.2}$$

$$\sum_{a \in \mathcal{A}} t(a, I, T) x_a \leq Cost \quad \forall I \in \mathcal{I}' \tag{4.3}$$

In the above ILP, $y(a, I, T)$ and $t(a, I, T)$ are constants, and $Cost$, x_a are variables. The maximum cost of running the portfolio on any instance is minimized by (4.1) and (4.3). Inequality (4.2) guarantees that all instances in the training set are covered by the best solutions which can

be obtained by time T. Thus, we maximize the chance of covering a new (unseen) instance with the best possible solution because our portfolio is capable of covering all the training instances with the best solutions known, while minimizing the overall time cost of running the portfolio. Note that each algorithm in the portfolio constructed by CP (4.1)-(4.3) must cover uniquely at least one instance. Otherwise the algorithm can be removed, decreasing $Cost$ without violating (4.2). Note that if $t(a, I, T) = T$ for all instances and algorithms, then the objective of CP is to minimize the number of algorithms in the portfolio, which is equivalent to the **NP**-hard set cover problem.

If the computational effort necessary to run all the algorithms obtained from the above formulation is too high, then it is possible to weaken the objective and minimize the maximum distance from the best solution subject to computational cost that is acceptable. The relative distance of some solution quality from the quality of the best solution known by T will be referred to as *regret* for instance $I \in \mathcal{I}'$. Let $q(a, I, T) = \min_j\{q^j_{aI} : t^j_{aI} \leq T\}$ be the best quality score obtained by algorithm a on instance I by time T. Let decision variable u_{aI} equal to 1 denote that algorithm a is instance I solver in the portfolio and equal to 0 in the opposite situation. The following mixed integer linear program (MIP), referred to as REGRET PORTFOLIO (RP for short), minimizes the greatest regret under limited computational cost:

$$
\begin{aligned}
RP: \quad & \min G && && (4.4)\\
& \frac{q(a, I, T)}{q_{\min}(I, T)} u_{aI} \leq G && \forall I \in \mathcal{I}' \quad \forall a \in \mathcal{A} && (4.5)\\
& \sum_{a \in \mathcal{A}} u_{aI} \geq 1 && \forall I \in \mathcal{I}' && (4.6)\\
& x_a \geq u_{aI} && \forall I \in \mathcal{I}' \quad \forall a \in \mathcal{A} && (4.7)\\
& \sum_{a \in \mathcal{A}} t(a, I, T) x_a \leq Cost && \forall I \in \mathcal{I}' && (4.8)
\end{aligned}
$$

In the above MIP, $t(a, I, T), q(a, I, T), q_{\min}(I, T)$ and $Cost$ are constants, u_{aI}, x_a are binary variables, and G is a fractional variable. Fraction $q(a, I, T)/q_{\min}(I, T)$ in inequality (4.5) measures the regret of using algorithm a for instance I. Minimization of the biggest regret on any instance is guaranteed by (4.4) and (4.5). By inequality (4.6) each instance has an algorithm providing a solution. Inequality (4.7) ensures that algorithm a chosen as a solver for instance I is counted as a member of the portfolio. Inequality (4.8) guarantees that the algorithm portfolio runs on any

instance in runtime limited by $Cost$. The $Cost$ is considered constant in (4.4)-(4.8) in order to avoid a trade-off between computational cost and solution quality guarantees G. For practical reasons, in the following text $Cost$ will be a small multiple of T.

4.7 Building Algorithm Portfolios

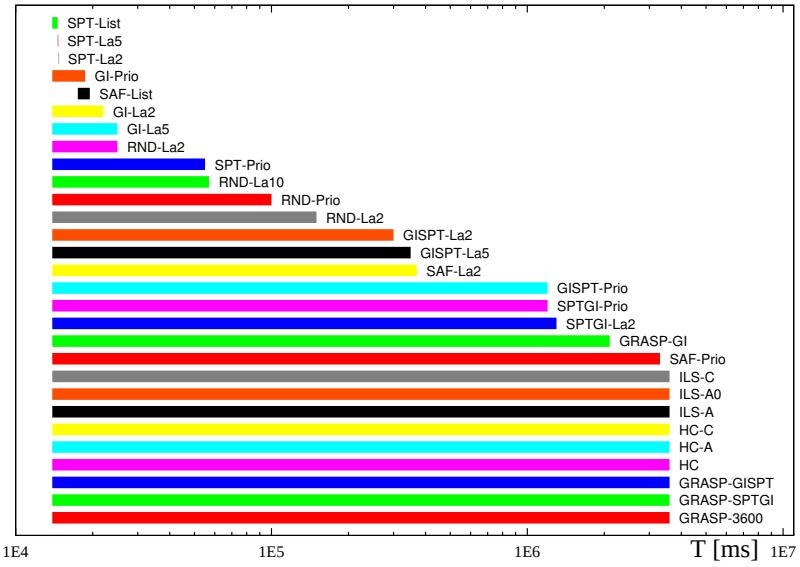
In this section we study the evolution of algorithm portfolios with changing runtime limit T. Unlike in Section 4.5, where algorithms were assessed individually, sets of algorithms are the subject of this section. The algorithm portfolios have been constructed for the four datasets introduced in Section 4.4. The complete results are collected in Appendix C.

4.7.1 Cover Portfolios

The ILP (4.1)-(4.3) constructing a cover portfolio for 1200 instances and 72 algorithms was solved by CPLEX 12.8 on a PC computer with Intel i7@2.8GHz CPU in less than 1s. Thus, the time needed for solving CP formulation of this size is not a limitation here. The evolution of the algorithm portfolios in time T for method CP can be seen in Figs 4.5 and 4.6. Runtime limit T is shown along the horizontal axis. The T values were selected with the resolution of 10ms for $T <$ 1s, 100ms for $T \in [1s, 10s)$, 1s for $T \in [10s, 100s)$, 10s for $T \in [100s, 1000s)$, 100s for $T > 1000s$. Runtime limits T span from the longest shortest runtime for any instance $\max_{I \in \mathcal{I}'} \min_{a \in \mathcal{A}} \{t^1_{aI}\}$, to the upper runtime limit of 3600s. Figs 4.5a and 4.6a show portfolio evolution in T. Figs 4.5b and 4.6b show the portfolio cost changes vs T. The bars in Figs 4.5a and 4.6a represent algorithm membership in the portfolio. The algorithms not included in the portfolio for any runtime limit T are omitted. The performance of portfolios is expressed as the number of included algorithms and $Cost$ of (4.1)-(4.3) expressed in T units (that is, $Cost$ normalized to T, Figs 4.5b and 4.6b).

To construct the portfolios presented in Fig.4.5, the instances in dataset 1 were used. Notice that, in general, simple greedy algorithms have limited capability to search the solution space but they are fast. Conversely, metaheuristics are more capable of searching the solution space, but their search takes time. Hence, greedy algorithms should be present in the portfolios for short runtime limits T and metaheuristics should dominate for large T. This effect indeed can be observed in Fig.4.5a and in time evolution of all other portfolios. However, a surprising effect can

a)



b)

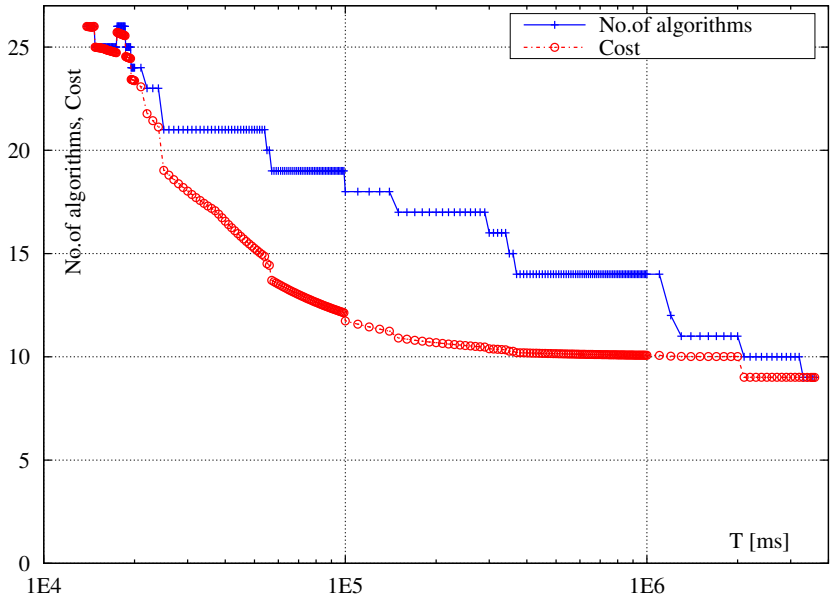
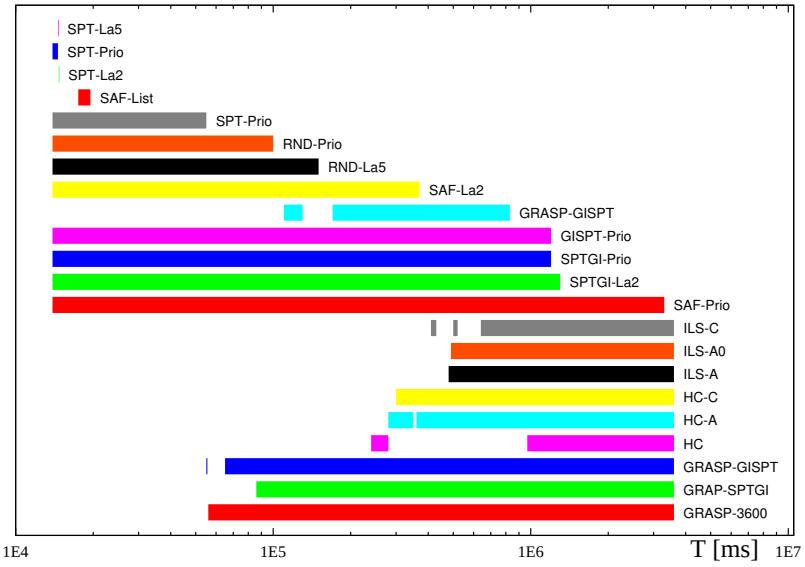


Figure 4.5: Cover portfolio built on all random instances N (dataset 1): a) portfolio evolution in time T, b) size and cost of portfolio (in T units), vs T.

be observed in Fig.4.5a: even for short runtime limits T time-costly metaheuristics are present in the portfolio. This can be explained in the following way: For n ranging from $n = 2$ to $n = 10000$ in dataset 1, algorithm runtimes are very dispersed. Large instances with $n = 10000$ determine the time $\max_{I\in\mathcal{I}'} \min_{a\in\mathcal{A}}\{t^1_{aI}\}$ in which any instance from the whole set $\mathcal{I}'$ can be solved by any algorithm. This runtime limit is large enough to allow solving small (e.g. $n = 2, 5$) instances by metaheuristics. Therefore, it is possible and profitable to keep universal algorithms like metaheuristics to cover some small instances and a few greedy algorithms to cover the remaining instances (including the large-n ones). With growing T, metaheuristics domination in providing the best solutions is growing and greedy algorithms are phased out from the portfolio. Consequently, the portfolio has a size of 26 algorithms for small T (see Fig.4.5b) and it decreases to 9 algorithms at T=3600s. Observe the presence of list and look-ahead algorithms which seemed inefficient in the evaluation conducted in Section 4.5. It means that these methods are useful in dealing with some special instances under certain runtime limits. Another observation is that even RND algorithms can be useful when runtime is limited. It can be concluded that the portfolio approach has better capability to discern algorithm advantages in limited runtime than the classic approaches analyzing central tendencies of algorithm performance. In Fig.4.5b, the portfolio performance score is expressed as the number of algorithms and $Cost$ in units of T. For a wide range of T values the number of algorithms is bigger than $Cost$. Since $Cost/T$ can be interpreted as the number of parallel processors required to run the portfolio, the number of algorithms exceeds the number of necessary processors. This means that most of the chosen algorithms have shorter runtimes than timeout T.

To construct the portfolios presented in Fig. 4.6, the instances in dataset 2 (all with $n = 10000$), were used. Hence, for each algorithm the runtimes are more concentrated. Consequently, the effect of hiding metaheuristic runtimes on small instances in $\max_{I\in\mathcal{I}'} \min_{a\in\mathcal{A}}\{t^1_{aI}\}$ set by a greedy algorithm on a large instance, described in the preceding paragraph, is not present. It can be verified in Fig.4.6 that for the smallest runtimes only greedy algorithms are present in the portfolio. With growing T, metaheuristics gradually join the portfolio, but without eliminating the greedy methods, and finally the greedy algorithms are phased out. Thus, the portfolio size grows from 8 to 13 algorithms (at 1000s) and then decreases to 9 metaheuristics (Fig.4.6b). This process is visible also in the $Cost$ of portfolio: initially only greedy algorithms with constant runtime (on instances $\mathcal{I}'$) are present in the portfolio and the cost in T units is decreasing

a)



b)

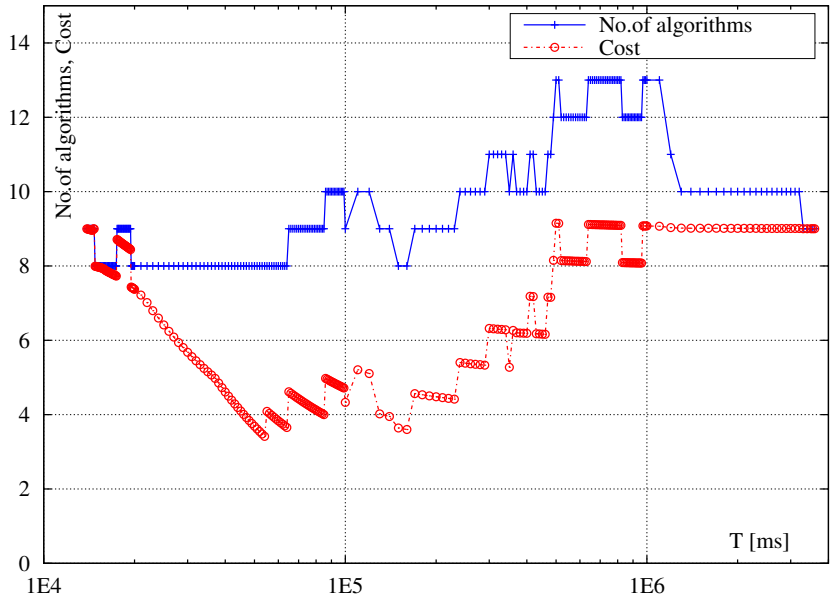


Figure 4.6: Cover portfolio built on random instances N for $n = 10000$ (dataset 2): a) portfolio evolution in time T, b) size and cost of portfolio (in T units) vs T.

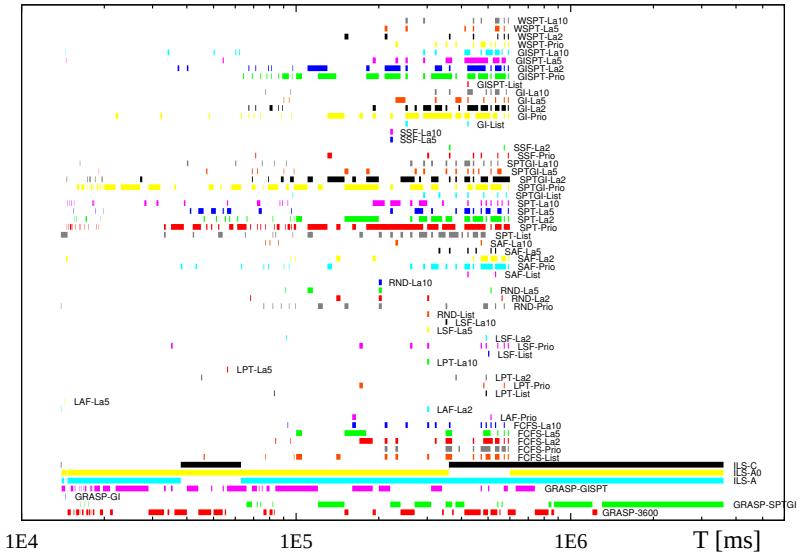
because T is increasing. At roughly $T = 56$s metaheuristics start joining the portfolio and the cost is increasing stepwise. For the random instances M (datasets 3, 4) similar behavior has been observed. Therefore, we do not report on these results here. The cover portfolios for datasets 3 and 4 can be found in Appendix C.

4.7.2 Regret Portfolios

Example evolution in T of regret portfolios constructed by RP formulation (4.4)-(4.8) is shown in Fig.4.7a for dataset 1 and $Cost = 4T$. Fig.4.7b shows the portfolio regret G score, size of the portfolio and the number of algorithms exchanged between neighboring evaluation points in T. Solving the RP formulation is more time consuming. For example, for dataset 1 ($|\mathcal{I}'| = 1200$ instances) and $|\mathcal{A}| = 72$ algorithms the median CPLEX runtime was 307s. For practical reasons, the MIP solver runtime was limited to 1800s and 7% of the MIPs used to generate Fig.4.7 stopped after 1800s. Consequently, for certain values of T, the portfolios may be sub-optimal solutions of the RP formulation. The set of algorithms present in the regret portfolios changes with T quite considerably (Fig.4.7a). The number of algorithms exchanged follows the number of algorithms in the portfolios. Both these numbers are especially big around $T = 600$s (Fig.4.7b). The big number of algorithms in the portfolio and the continuous exchange process is a result of the $Cost = 4T$ limit. Note that the costs of cover portfolios in Fig.4.5b are bigger than $4T$. As a result, the regret portfolios have to exchange the algorithms intensively to minimize regret G (Fig.4.7b). Bigger T allows for running more algorithms and the size of the portfolio increases. Note that three ILS-type metaheuristics in the regret portfolio for $T > 600$s start from the best solution built by any greedy algorithm. The $T = 600$s time budget is sufficient to execute this initial step and all the greedy algorithms become dominated. Consequently, runtime limits $T > 600$s are sufficiently large to allow restricting the portfolio to 4 (as $Cost = 4T$) winning metaheuristics. The phenomena observed in the portfolios for other $Cost$ limits and other datasets are similar, so we do not discuss them here. See Appendix C for these portfolios.

The regret G score and the number of algorithms in the portfolio for other values of the $Cost$ limit are shown in Fig.4.8. Notice that with the increasing $Cost$ limit the regret score G improves. The number of algorithms in the portfolio has a more complex behavior: At both ends of the time range the number of algorithms is equal to $Cost/T$. At short Ts, runtime limit is so restricted that only $Cost/T$ algorithms can be executed. For large Ts, there are

a)



b)

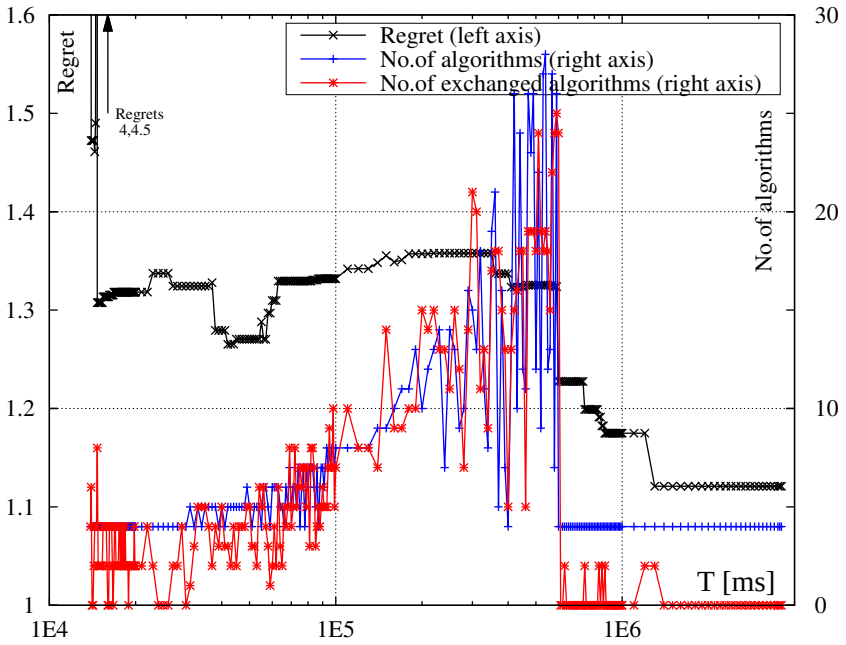
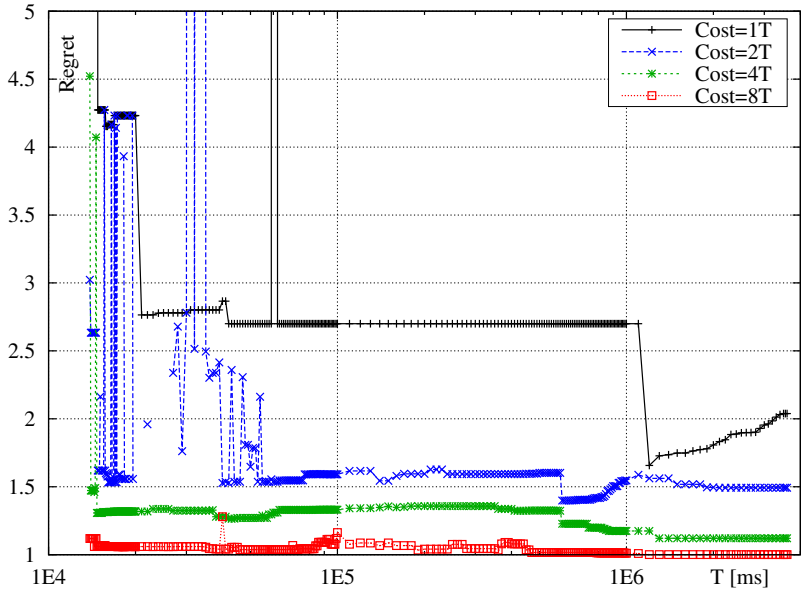


Figure 4.7: Regret portfolio built for all random instances N (dataset 1), $Cost = 4T$: a) portfolio evolution in time T, b) number of algorithms in portfolio, number of exchanged algorithms (right axis), regret values (left axis) vs T.

a)



b)

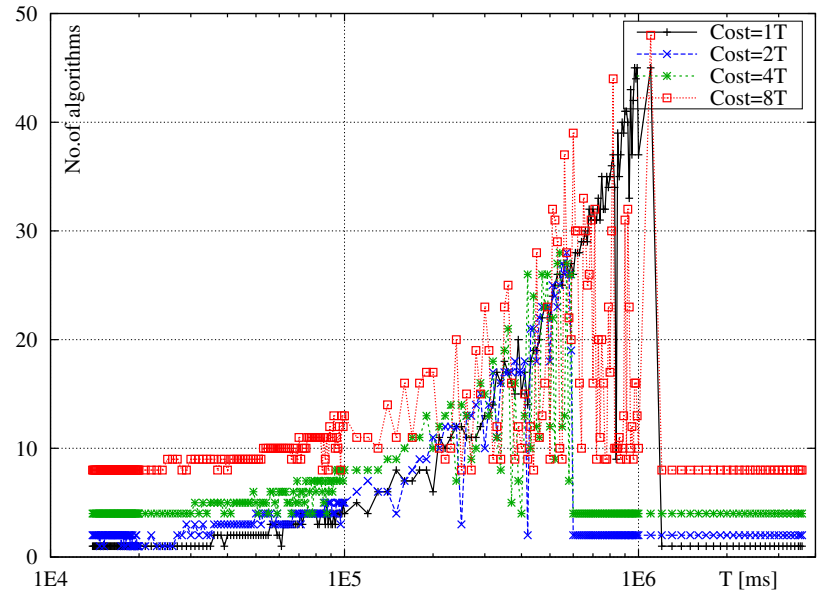


Figure 4.8: Scores of regret portfolios built on all random instances N (dataset 1): a) regret G score, b) number of algorithms in portfolio, vs time T and $Cost$.

$Cost/T$ dominating metaheuristics which stop computation at runtime limit T. In between, no algorithm dominates consistently, and the portfolio intensively exchanges the algorithms with T. Unless $Cost$ limits are sufficiently large, the volatility and the large number of algorithms restrict the practical applicability of regret portfolios.

Finally, notice that portfolios converge with T. For small T the portfolios differ and comprise diversified sets of greedy algorithms. But with increasing T the portfolios converge to a common subset of universal algorithms.

4.8 Performance of Algorithm Portfolios

In this section we evaluate performance of the algorithm portfolios. Firstly, the portfolios developed for one dataset (training) were used to solve instances from some other dataset (validation). In this way we verify portability of the portfolios, i.e., their capability to solve new (unseen) instances. To this end, for each tested runtime limit T, we used a portfolio developed for the training dataset to solve instances from the validation dataset. The portfolio results were compared against the solutions obtained by all our algorithms on the validation datasets. The number of the portfolio wins (i.e. instances solved to the best solution known at the considered T) and the relative distance from the best solution known by T, $q_{\min}(I,T)$, are measures of the portfolio quality. If none of the portfolio algorithms is able to construct a solution by T, then $q_{\min}(I,T)$ is undefined. For practical purposes, $q_{\min}(I,T)$ in such cases was assigned a very large value of $MWFT$. In order to avoid developing and testing portfolios on the same data, random instances M were used as validation datasets for training random instances N, and vice versa. Secondly, we compare solution quality and cost of the cover portfolios against simpler portfolios.

4.8.1 Cross-Validation on Random Instances

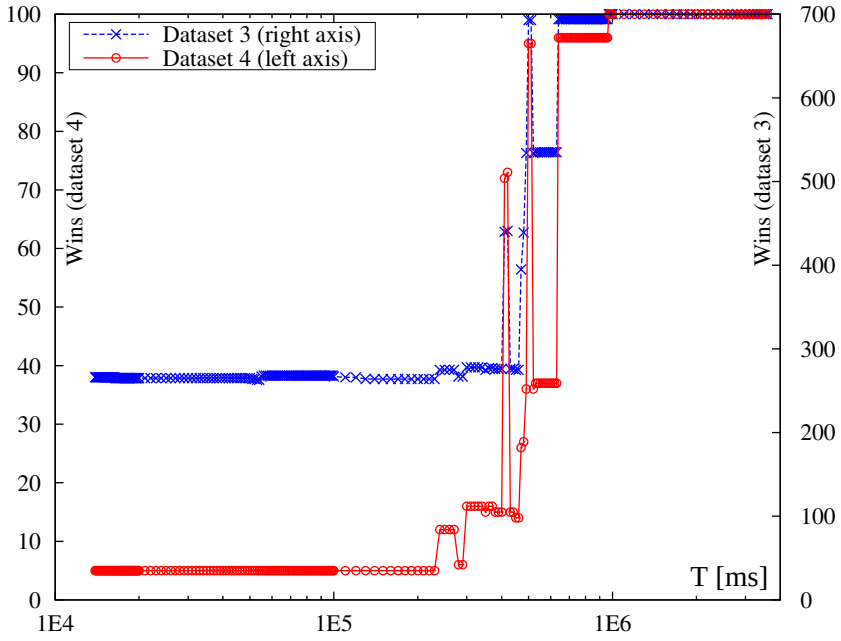
Cover portfolios. Cover portfolios built on training dataset 1 (Fig.4.5) returned all the best know solutions for all runtime limits T for validation datasets 3 and 4. On the one hand, these portfolios perform very well and are robust with respect to solution quality. On the other hand, they comprise more algorithms than, e.g., the portfolio built for $n = 10000$, and possibly the computational costs are higher than necessary. Nevertheless, these portfolios are a subset of all

available algorithms and there is some gain on computational cost.

Performance of the cover portfolios developed for random instances N with $n = 10000$ (training dataset 2) on all random instances M (validation datasets 3 and 4) is shown in Fig.4.9. At runtime limit $T = 970$s all instances (700 instances in dataset 3, 100 instances in dataset 4) are covered with the best solutions (Fig.4.9a). The median of the solution quality ($MWFT$) relative distance from the best solution that could be obtained by T is below 2% for dataset 3 and below 10% for dataset 4. The worst-case distance from the best known solution (quartile Q4) is below 270% (dataset 3, Fig.4.9b). There is a switch in performance around 400–700s, resulting from including metaheuristics in the portfolio. Instances in dataset 2 are relatively hard compared to instances in datasets 3, 4, and some metaheuristics are switched on only for $T > 400$s (see Fig.4.6). Still, it can be concluded that the portfolio performs well because it either delivers solutions with the $MWFT$ close to the best known, which is the case for $T < 970$s, or builds the best known solutions, when runtime limits are higher.

Performance of the cover portfolios constructed for random instances M (training dataset 3 and 4) on random instances N (validation datasets 1 and 2) is shown in Fig.4.10a. As mentioned, there are instances in datasets 1 and 2 which require 13.9s to be solved by any algorithm. Consequently, the number of wins in the whole dataset can be compared only for $T > 13.9$s. There are also cases for $T > 13.9s$ that no solution is provided by an algorithm in the portfolio. The $MWFT$ quality score is undefined in such a situation and it is not possible to calculate the statistics of distance from the best known solution in such cases. Hence, only the number of wins is shown in Fig.4.10a. For validation dataset 2 (random instances $N, n = 10000$) the cover portfolios developed for random instances M (datasets 3 and 4), for $T \in [13s, 300s]$ did not deliver any solution. This is a result of the fact that the portfolios built on datasets 3 and 4 give up using greedy algorithms around $T = 10$s in favor of metaheuristics (see Appendix C). However, only greedy algorithms are able to solve the instances with $n = 10000$ in validation datasets 1, 2 at $T \approx 13.9s$. Hence, the disparity between the hardness of the training and the validation datasets results in inferior performance. There are also instances with $n < 10000$ in validation dataset 1 (all random instances N) which are solved in time shorter than 10s by the portfolio algorithms. For this reason the number of wins for validation dataset 1 and small values of T is approximately 900. For both validation datasets the number of wins is growing with T.

a)



b)

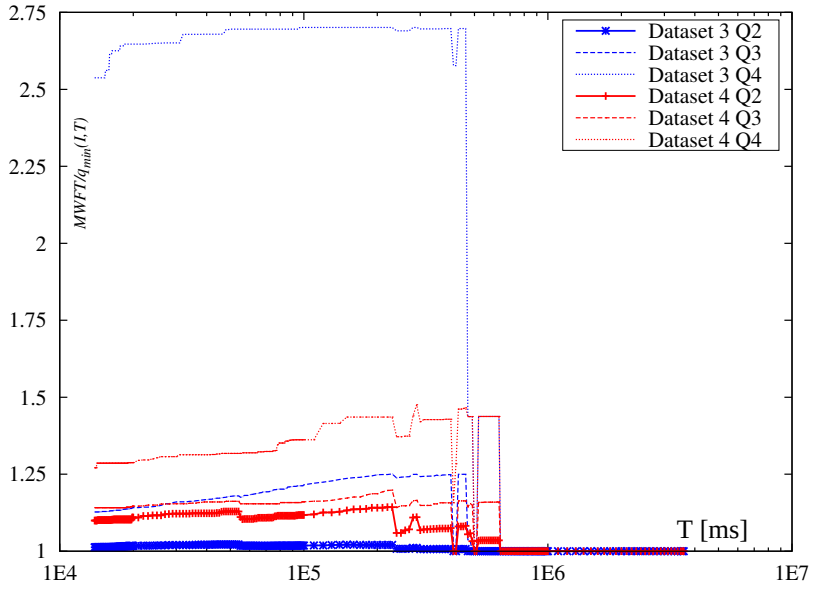
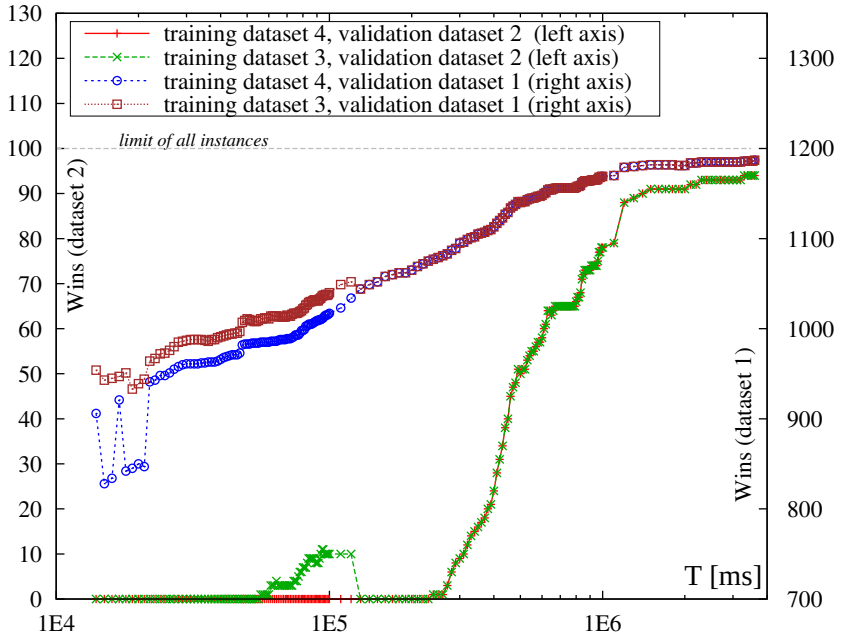


Figure 4.9: Validation of the cover portfolio built for random instances N with $n = 10000$ (training dataset 2) on all random instances M (validation datasets 3,4): a) number of wins, b) relative distance from the best known solution, quartiles Q2, Q3, Q4.

a)



b)

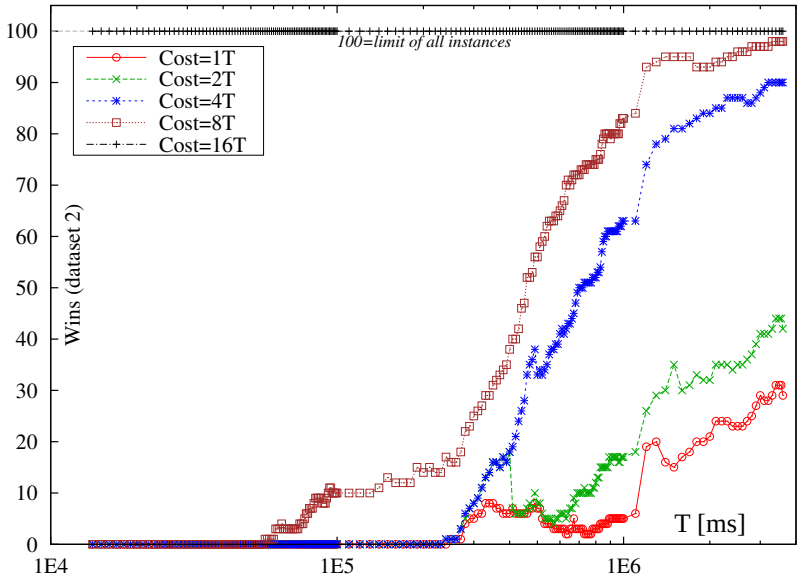


Figure 4.10: a) Wins of the cover portfolios built for random instances M (training datasets 3,4) on random instances N (validation datasets 1,2). b) Wins of the regret portfolios built for random instances $M, m = 2$ (training dataset 4) on random instances $N, n = 10000$ (validation dataset 2).

Regret portfolios. Similar phenomena can be observed in the regret portfolios. Yet, one more impacting factor is the *Cost* allowance (inequality (4.8)). With growing *Cost* limit, more computationally costly algorithms can be included in the regret portfolio at lower runtime limits T. For example, the number of wins of the regret portfolio built for training dataset 4 ($m = 2$) on the validation dataset 2 ($n = 10000$) is shown in Fig.4.10b. On the right side of the picture the number of wins is growing with growing *Cost* limit.

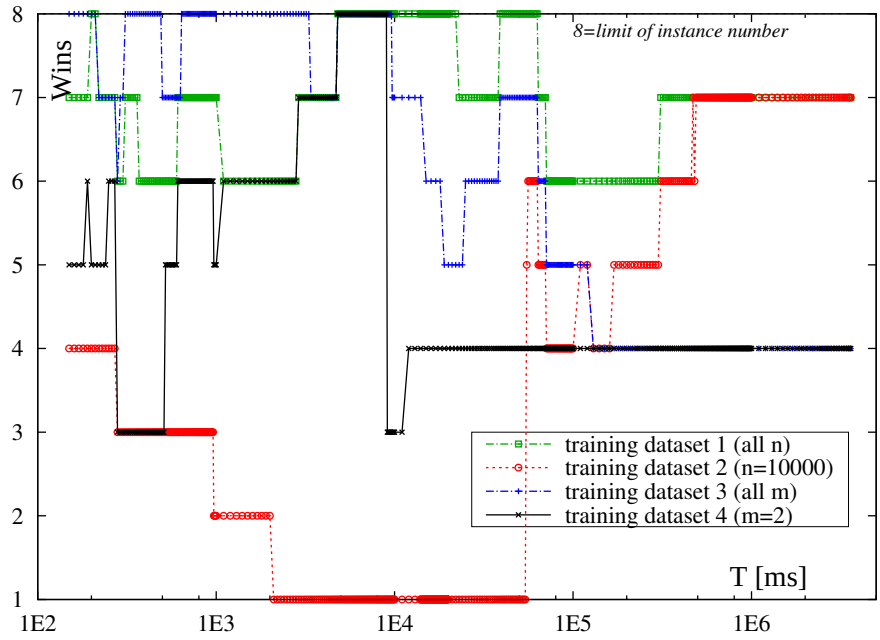
4.8.2 Cross-Validation on Real Instances

Cover portfolios. Performance of the cover portfolios constructed for datasets 1–4 (random instances N and M) on the dataset of 8 real instances varied depending on the runtime limit T (see Fig.4.11a). In the cases when the portfolio did not win, two behaviors were observed: either no solution was found by T, or the solutions were very close to the best one found by T. The former was the case of the portfolios built for datasets 3 and 4 (random instances M), but it was not the case of portfolios derived from datasets 1,2 (random instances N). If a solution was found, but it was not the best known by T, then the relative distance from the best one known by T did not exceed 0.03%.

Regret portfolios. Performance of the regret portfolios on real instances was similar to the performance of cover portfolios. On top of this, the *Cost* limit (inequality (4.8)) imposed restrictions on the size of the portfolio and consequently on its capability to deliver good solutions. Hence, with growing *Cost* limit, the quality of the portfolio solutions improved (see Fig.4.11b). The number of wins of the regret portfolios on real instances, again depends on T. In the cases when a solution was found, performance of the regret portfolios in terms of the relative distance from the best solution obtained by time T for almost all tested portfolios and Ts was below 0.1%. There were four exceptions: for datasets 1,2, $Cost = 1T$ and $T \in [14s, 14.6s]$ the relative distance was 207 times, for dataset 2, $Cost = 1T$ and $T \in [150s, 160s]$ the relative distance was 52%, and for dataset 3, $Cost = 1T$ and $T \in [1.3s, 2.6s]$ the relative distance was 2.7%.

Overall, it can be concluded that the performance of the portfolios on real instances was satisfactory.

a)



b)

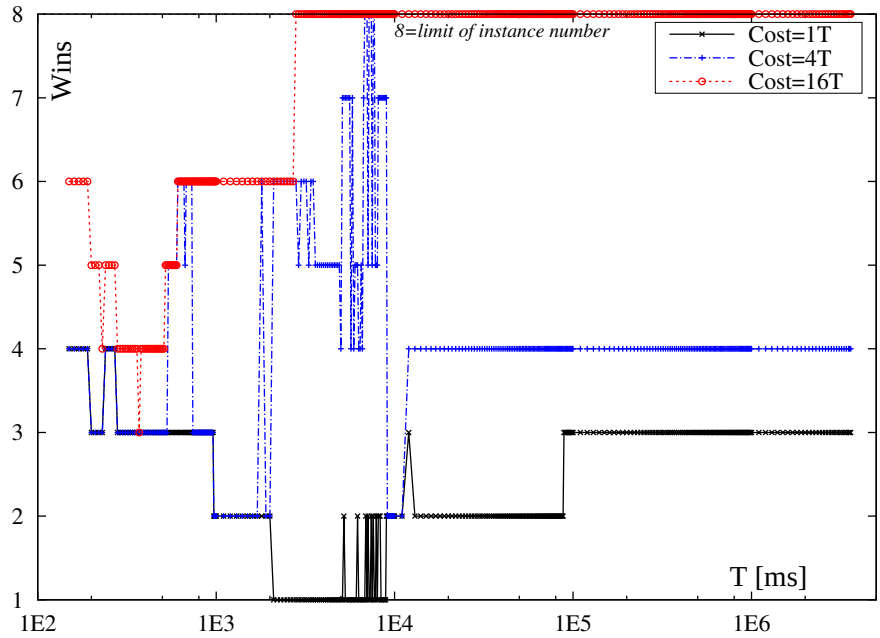


Figure 4.11: Performance of algorithm portfolios on real instances: a) number of wins for cover portfolios, b) number of wins for regret portfolios built for random instances $M, m = 2$ (dataset 3).

4.8.3 Comparison with Fixed Portfolios

The above evaluation considered portability of portfolios between training and new (unseen) datasets, but it did not tell on the gains from using the portfolios. The gains may be twofold: in quality of the solutions and in the computational cost. Furthermore, there can be other portfolios than CP (4.1)-(4.3) or RP (4.4)-(4.8). In this section we compare the number of wins and computational cost of the cover portfolios against four alternatives: 1) *FixPF1* – a portfolio of ILS-A, FCFS-List, SAF-Prio; 2) *FixPF2* – portfolio: ILS-A0, HC-C, GRASP-3600, SG, comprising the best different type metaheuristics in the central panel of Table 4.2 (relative distance from the best solution) and SG; 3) *FixFAT4* – the first four methods in the right panel of Table 4.2 (unique wins); 4) *SG* – Super Greedy which is a portfolio itself. Since algorithm membership in the above portfolios is independent of the runtime limit T, we will refer to them as to *fixed* portfolios.

Performance of cover and fixed portfolios on dataset 1 (all instances N) is compared in Fig.4.12. The results for other datasets and for regret portfolios are similar, so we do not report on them. In Fig.4.12a the number of wins in dataset 1 is shown. By definition, the cover portfolios built for this dataset have all 1200 wins (denoted as CP for dataset 1). The cover portfolios developed on training instances M are the second-best with a few exceptions at $T < 20$s. The number of SG wins is constantly decreasing with the increasing T. Since dataset 1 has quite diversified sizes of instances, the number of SG wins stabilizes around 480, but for dataset 2 ($n = 10000$) the number of SG wins decreases to 0 at $T = 3600$s (not shown here). Similarly, for FixPF2 the number of wins decreases to 770 at $T = 3600$s because, on the one hand FixPF2 comprises SG, but on the other hand it also has three metaheuristics. The simplest fixed portfolio FixPF1 loses against SG for small values of T because ILS-A is too costly, and FCFS-List, SAF-Prio are not the greedy algorithms providing the biggest number of wins at these values of T. Conversely, SG loses against FixPF1 for larger T because FixPF1 comprises one metaheuristic, while SG is a collection of greedy algorithms.

In Fig.4.12b computational cost in T units is shown. It can be seen that for small values of T the cost of SG and FixPF2 is unnecessarily large because some unneeded greedy algorithms are executed. Since the runtime of greedy algorithms is constant and the computational cost is expressed in T units, the cost of SG and FixPF2 as shown in Fig.4.12b decreases with increasing T. For small runtime limits T the computational cost of the cover portfolio winning on all 1200

test instances (denoted as CP for dataset 1) is smaller than the cost of SG and FixPF2 because only algorithms winning on some instances are included in CP. The cost of CP for dataset 1 is also bigger than for the other cover portfolios developed for instances M (denoted CP for dataset 3 and 4) and the cost of FixFAT4, because all 1200 instances are covered by the CP for dataset 1. Comparing the wins of CP for dataset 1 with the wins of CP for instances M and the wins of FixFAT (Fig.4.12a) against computational costs of these portfolios (Fig.4.12b) it can be concluded that covering a few instances with the best solutions (which CP for dataset 1 covers and, e.g., FixFAT4 does not) is quite costly computationally.

It can be concluded that cover portfolios have advantage in the number of wins over the fixed portfolios. With respect to the computational cost, cover portfolios are much more effective than collections of greedy algorithms like SG. For large time limits T the cover portfolios bear inevitable costs of covering all instances even the hardest ones.

Let us summarize this section with some observations on developing algorithm portfolios. Section 4.8.3 shows that fixed portfolios are not competitive for all time budgets T, when compared with the portfolios evolving with T. It is prudent to construct algorithm portfolios using datasets similar to the target application. Datasets with diversified hardness (as dataset 1) result in algorithm portfolios which may be large yet are robust with respect to the quality of constructed solutions. It is also reasonable to develop portfolios on hard instances (like dataset 2) because these portfolios will include fast greedy algorithms for short runtimes and universal metaheuristics for long runtimes. Constructing portfolios on datasets of significantly lower computational hardness (as datasets 3 and 4) than the target production instances is not recommended because the algorithm runtime ranges in the training and in the target applications will not match, resulting in solutions of inferior quality.

4.9 Summary of Algorithm Selection

In this chapter we considered the problem of selecting algorithms to solve large BAP problems with time budget limitations. This study was dictated by a practical need for simulating port ship traffic where the simulation time is an important decision variable. We tested 72 fast heuristics. Individual evaluation of these heuristics led to a rather expected conclusion that certain greedy algorithms (SPT, SPTGI, GISPT, GI) are the best for short time budgets, and

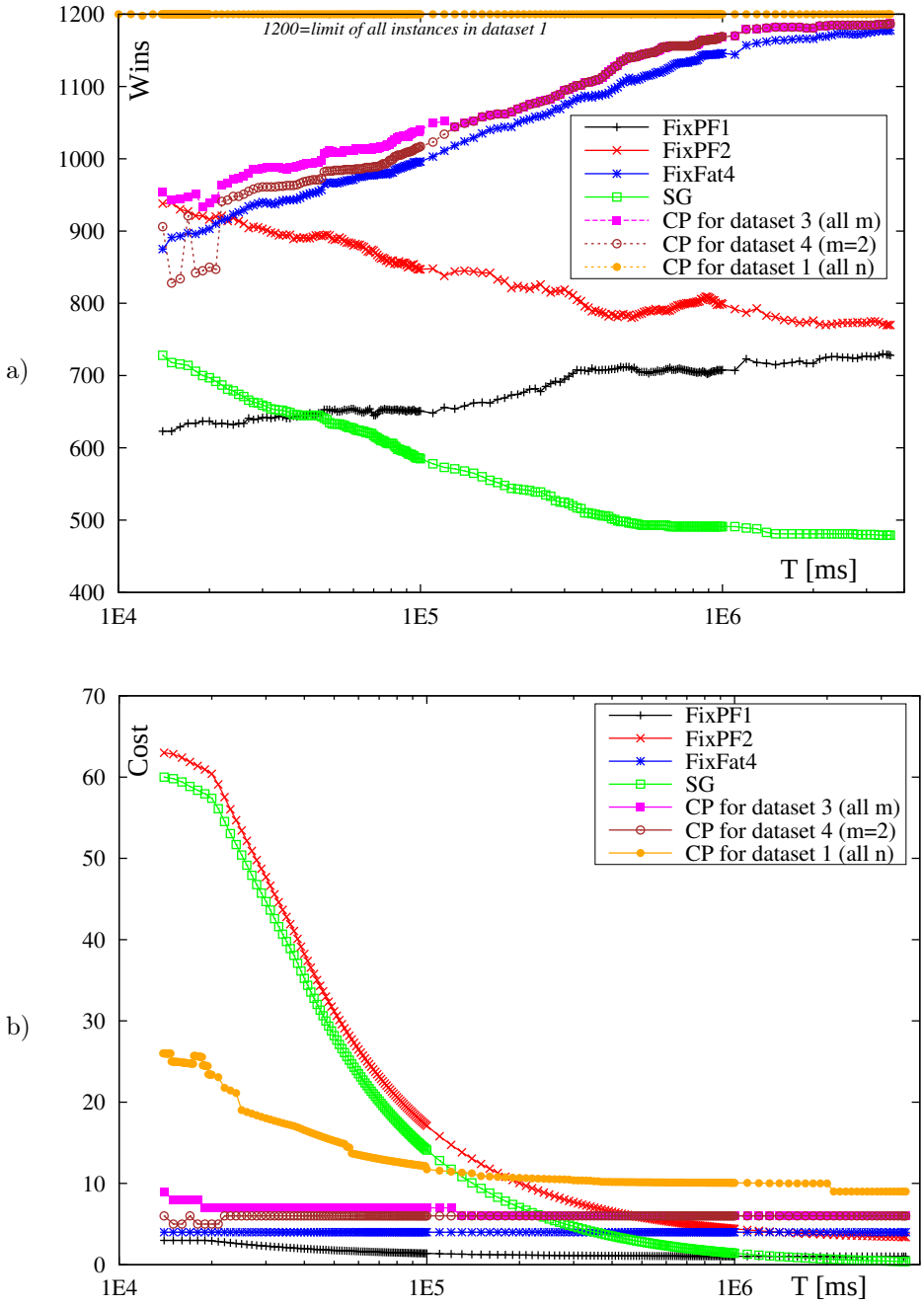


Figure 4.12: Comparison of algorithm portfolios against fixed portfolios on dataset 1 (all instances N). a) number of wins, b) computational cost.

metaheuristics (HC-C, ILS-C, ILS-A) are the best for large time budgets. Then we turned to the concept of algorithm portfolios for solving BAP. In this way we departed from the common idea of "one best" algorithm to solve all instances of the problem, in favor of sets of algorithms applicable under different runtime limits. The portfolios are built in such a way that training instances are covered with the best solutions found within the runtime limit, or with the solutions minimizing quality loss, while the cost of running the chosen algorithms is minimized. The study of portfolio evolution in time not only confirmed the earlier qualitative intuition that fast greedy algorithms should be used for short runtime limits and metaheuristics for long limits, but also provided a quantitative way of traversing from the former set of algorithms to the latter.

At short runtime limits computational cost of the portfolios is smaller than for a straightforward collection of greedy algorithms. At large runtime limits the selected set of metaheuristics covers with wins a larger set of instances. The portfolios were shown to be better at recognizing that certain algorithms (though weak on average) are effective in solving certain instances under limited runtime. Algorithm portfolios evolving in time provide a method consistently recognizing useful algorithms on the basis of their performance. No algorithm is excluded from a portfolio as long as it is able to build best solutions for any instance under some runtime limit. Considering the obtained portfolios, it can even be claimed that the classic algorithm evaluation methods based on the central tendencies are myopic. We have also given recommendations on constructing training instances for developing effective algorithm portfolios. Our approach has been successfully applied to realistic instances built from real port data.

We believe that the study presented in this chapter may be interesting from the methodological point of view because a non-standard approach to the evaluation of heuristics involving the runtime-quality trade-off has been proposed. Further research should go beyond the limitations of this study and address, e.g., the following subjects: (1) Adding other advanced metaheuristics to the portfolios. (2) Extending this approach to other BAP formulations. (3) Studying how to construct training instances for the portfolio robustness, low computational cost, and high solution quality. (4) Applying the approach developed here to other combinatorial optimization problems. (5) Automating the process of test instance generation and selecting the algorithm portfolios so that the human expert is taken out of the decision process. (6) Finally, the selected algorithm portfolios will be exploited in long-term port simulation considered in Chapter 6.

Chapter 5

Container Ship Traffic Model for Simulation Studies

5.1 Ship Traffic Model

In this chapter, a container ship traffic model (STM) for port simulation studies is developed. Consider classic operations research and combinatorial optimization problems which take advantage of existing benchmark datasets. For example, there are E.Taillard benchmarks for shop scheduling problems [82], Parallel Workload Archives for supercomputer jobs scheduling [24], NEO instances for vehicle routing problem [32], and even on a broader scope TSPLIB [69], OR-Library [3] benchmark collections. Port logistics problems also need benchmark instances to test algorithm runtime and their solutions quality. Until recently there was limited availability of such data. The aim of this thesis is to develop a model of container ship traffic. Such a model is vital in terminal design analyses, and testing performance of optimization algorithms. This kind of research requires accurate information about the ship stream to build test scenarios and benchmark instances. A statistical model of ship traffic is developed on the basis of container ship arrivals in eight world ports. The model provides three parameters of the arriving ships: ship size, arrival time and service time. The stream of ships is divided into classes according to vessel sizes. For each class, service time distributions and mixes of return time distributions are provided. A model of aperiodic arrivals is also proposed. The results obtained are used to

Table 5.1: Summary of notations for Chapter 5

a_i	number of ships of class i in some port
$(c_{i-1}, c_i]$	interval of ship lengths in cluster (size class) i
η_a	fraction of aperiodic arrivals in a certain port, or terminal
ι	intensity of traffic, an average number of arrivals in some time interval, e.g. in arrivals per week [a/w]
k	number of ship classes
L_j	length of ship j
$\mathcal{L}_i$	ship length model for cluster i in some port
ν_j	number of ship j calls at a port in the historic dataset
n	number of vessels in some port as physical objects
N	number of calls at a port (sum over vessel calls)
p_j	processing time of ship j (one of possibly several instantiations in historic dataset and in STM)
$\mathcal{P}_i$	model of ship processing times for cluster i in some port
r_j	arrival (ready) time of ship j (one of possibly several instantiations in historic dataset and in STM)
ρ_j	return time of ship j, that is duration between two arrivals of the same ship (one of possibly several instantiations in historic dataset and in STM)
$\mathcal{R}_i$	model of return times for ship size class i

compare port specific features.

Further organization of this chapter is the following. The rationale behind the STM is presented in Section 5.2. Section 5.3 is dedicated to the method of ship traffic model construction. In Section 5.4, we outline how to use the STM to generate ship stream data. In Section 5.5 ports are compared using relationships in the analyzed data. Section 5.6 summarizes results of this chapter. The notations are collected in Tab. 5.1. Due to the size of collected results, only selected examples are presented in the main text of the thesis. The STM parameters are provided in Appendix D. Benchmark instances built according to our STM are collected at [89].[1]

5.2 Features of Ship Traffic Model

In this section, the rationale behind the proposed ship traffic model and its relation with port logistics are presented. We will often refer to the example of BAP, which is both a central problem of port logistics and tightly connected with the QPP (cf. Section 2.3). Before discussing the key STM parameters, consider a simple example: A quay has one 500m-long berth. Vessel

[1] Due to the rules of access we are not allowed to release the source data.

Table 5.2: Example vessel data

vessel	A	B	C	D	E	F	G
p_j[h]	2	2	2	3	3	3	3
L_j[m]	110	120	130	390	380	370	400
Scenario 1							
r_j[h]	0	3	6	0	3	6	9
Scenario 2							
r_j[h]	0	0	0	0	0	0	0
Scenario 3							
r_j[h]	10	5	0	5	0	11	10

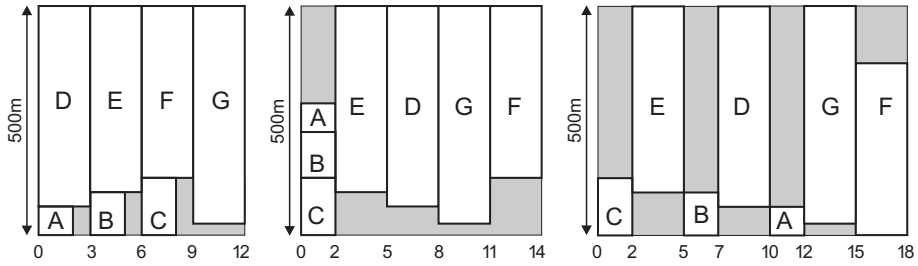


Figure 5.1: Example schedules. a) Scenario 1, b) scenario 2, c) scenario 3. Vessel and quay lengths are not proportional.

parameters are provided in Tab. 5.2. There are three scenarios of arrival process. Schedules for the three scenarios are shown in Fig. 5.1. The first scenario is simple to manage because vessels arrive in pairs fitting quay length and it suffices to serve the ships as they arrive. In the second scenario all vessels are ready simultaneously and to serve them a decision of combinatorial nature is needed. Namely, a sequence of vessel service has to be determined. The third scenario is adverse because vessels arrive in mutually incompatible pairs which results in low quay utilization. The scenarios that appear in reality may be neither so easy nor that adverse. Moreover, just vessel arrival process was tackled in this example. Other vessel parameters, port characteristics, the considered optimization problem open ways for more complex relationships to emerge. Hence, there is a need for distinguishing realistic requirements on a ship traffic model. In the remaining part of this section, we discuss three main parameters defining ship traffic.

The arriving ship j has a certain length L_j and must be given sufficient space along a finite quay. Large container mother ships travel long distances, e.g., on Asia-Europe or Asia-America lines, while small ships (feeders) operate on local connections. Consequently, large vessels call at the ports less frequently than the small ones. Hence, the ship length is a basic feature in an STM, and is highly related to the ship arrival time.

Temporal constraints on serving a ship are imposed by ship j arrival time r_j and ship service time p_j, also called processing time in this thesis. In the papers on BAP as a scheduling problem, each ship arrival is often considered as an independent event. But a single vessel may call at some port many times with some return intervals. In that case, the vessel will be called *returning*, or *periodic*. Otherwise it is called a *non-returning* ship, or a ship with *aperiodic* arrival. When a distinction between one arrival of a ship and the vessel as a physical object is important, we will use terms arrival, occurrence vs a physical vessel/ship. Ship ready times r_j may be realizations of some return process. In this case, the physical ship j return time will be denoted ρ_j.

The vessel processing times p_j may be a proxy to the number of loaded and unloaded TEUs assuming that the transfer rate is roughly constant. The processing times depend on berthing process, quay crane schedule, yard management, intra-port transport. Let paper [75] serve as an example of 1) port operations modeling, 2) complexity of the relationships, 3) problem sizes that can be currently solved. Instances with barely 48 vessels arriving over 7 days, on several terminals, with intra- and inter-terminal transport were optimized. This size of instances, appropriate for tactical decisions, are far from sufficient to represent details of alternative container terminal designs at the long-term strategic level planning, with consequences spanning over several year time horizons. Thus, instead of building a planning and scheduling representation of all elements of a container terminal it is simpler to use realistic vessel processing times (again, from the real data now available) with adequately represented dispersion. Hence, ship processing time distributions are essential components of the model.

A majority of port logistic formulations assume a deterministic approach. This means that parameters L_j, r_j, p_j are given numbers. However, it is hard to accept that r_j, p_j can be fixed in advance over time horizons typical of strategic planning. Ship arrival and processing times depend on many factors which cannot be controlled, and it is more plausible to accept that uncertainty in these values exists. For instance, more than 49% of vessels can be a day late and average deviation from estimated arrival time can exceed 2 days [55]. Hence, despite the

advertised schedules of liner companies, arrival times of periodic ships are not deterministic and ship traffic model should be considered as a stochastic process which samples objects (L_j, r_j, p_j).

Let us return to the interrelations between L_j, r_j, p_j parameters. The service given to a big ship has bigger value for the terminal operator than the service given to a smaller ship. Hence, there are stronger financial incentives to stick to the schedules planned for the big ships. This can be reflected in different dispersions of ship processing and arrival times. As already said, ship return and processing times depend on the ship sizes. In order to deal with such dissimilarity, different ship length *classes*, or equivalently called *clusters*, will be considered. Each cluster should have its own distribution of p_j and r_j values.

Finally, let us note that in the intended STM, we want to identify and generalize traffic patterns important to long-time strategic planning. It implies that short-term weekly and daily data patterns are less interesting as typical of contemporary operational optimization. We will return to this issue in Section 5.3.5.

5.3 Building of Ship Traffic Model

The ship traffic model is defined by the following elements: 1. ship class definitions, 2. model $\mathcal{L}_i$ of ship lengths for each class i, 3. model of processing times $\mathcal{P}_i$ for each class i, 4. ready time models: 4a. return time models $\mathcal{R}_i$ for each class i of returning ships, 4b. ready time model for non-returning ships. Thus, the method to build a ship traffic model on the basis of the actual data consists of three steps: 1. partitioning of the ships into size classes, 2. setting processing time distributions for each size class, 3. building return, or ready time models. In the next subsection, the historical dataset is described. Then, the STM parts are derived. After proposing components of the STM, a discussion of the model limitations and alternative approaches will be conducted.

5.3.1 The Dataset

Ship traffic model was built from the historical automatic identification system (AIS) data gathered in eight ports in the world in 2016. The dataset comprises a range of port sizes, from small traffic ports (Gdańsk) to the largest in the world (Singapore). Basic dataset information is collected in Tab. 5.3. The table comprises the number of calls at the ports, the number of physical

Table 5.3: Basic dataset information

port	Gdańsk	Long Beach	Los Angeles	Le Havre
number of calls N	471	995	1308	2271
physical ships n	81	242	309	559
returning ships	57	188	238	417
No.of unique L_js	31	65	61	105
η_a[%]	5.10	5.43	5.43	6.25
port	Hamburg	Rotterdam	Shanghai	Singapore
number of calls N	3294	3998	11606	18494
physical ships n	586	311	1233	1857
returning ships	466	240	1038	1634
No.of unique L_js	101	80	148	169
η_a[%]	3.64	1.78	1.68	1.21

vessels, returning vessels, the number of physical vessel lengths given with 1m resolution, and fraction η_a of aperiodic ships in the total number of arrivals. For example, in Gdańsk only 24 calls out of 471 were single visits. The fraction of non-returning calls ranges from 1.21% (Singapore), to 6.25% (Le Havre). Tab. 5.3 provides the first qualitative conclusion for port design and port logistic algorithm testing: a large majority of the vessels are returning. Consequently, the cardinality of physical ship set is rather limited.

5.3.2 Ship Size Clustering

The way of exploiting a ship and handling it at the terminal depends on its size L_j. Despite attempts to limit the number of ship classes (see Panamax, NewPanamax and so on), there is a large variety of vessel sizes (cf. number of unique L_js in Tab.5.3, and examples in Fig.5.2), but for practical reasons it is more convenient to use a few vessel size classes. Indeed, the ship traffic model dedicated to a set of similar ships is simpler, easier to develop and more accurate than a general model encompassing all possible ship lengths. We will show in Section 5.3.4 that a single size-general model for ship processing times p_j appeared unsatisfactory. Hence, vessels calling at a particular port were clustered according to their lengths.

The definition of a ship class (cluster) i consists in the ship size range. Then, number a_i of arrivals in the class can be calculated. Let $(c_{i-1}, c_i]$ be the range of vessel sizes for the ith cluster, where $c_0 = 0$ and $c_k = \max_j\{L_j\}$ for the last cluster k. The quality of vessel j fit in

Table 5.4: Normalized clustering quality scores

port	Gdańsk	Long Beach	Los Angeles	Le Havre
$k = 5$	2.10	1.88	1.57	1.91
$k = 6$	1.45	1.33	1.15	1.46
$k = 7$	1	1	1	1
port	Hamburg	Rotterdam	Shanghai	Singapore
$k = 5$	1.91	1.50	1.42	1.52
$k = 6$	1.09	1.25	1.06	1.31
$k = 7$	1	1	1	1

cluster i is measured by the distance:

$$d(j,i) = \begin{cases} \infty & \text{for } L_j > c_i \\ c_i - L_j & \text{for } L_j \leq c_i \end{cases} \tag{5.1}$$

Let $\mathbb{1}(j,i) = 1$ if $d(j,i)$ is the minimum for ship j and cluster i, and 0 otherwise. The quality of clustering is measured by the sum of all ship distances weighted by the number of ship j calls ν_j at the port considered:

$$Q(c_1, c_2, \ldots, c_k) = \sum_{j=1}^{n} \nu_j \mathbb{1}(j,i) d(j,i) \tag{5.2}$$

While setting the clusters, their number k and range ends c_i are the decision variables. Note that since the quality of clustering is weighted by ν_j, some groups of ships with many returning calls may be split into clusters of narrower range $(c_{i-1}, c_i]$, while some less frequent groups of ships may be merged together. The c_i values were calculated by generalized reduced gradient method [52] for $k = 5, 6, 7$. Values of the clustering quality measured by (5.2), normalized to the best result, are shown in Tab.5.4. Since the quality of clustering does not improve much by raising k from 6 to 7, and since using many ship size classes is unwieldy, we decided to limit the number of ship sizes clusters to $k = 7$.

The ends c_i of the size intervals for the ports considered are given in Tab.5.5. Additionally, the total number a_i of calls for cluster i and the number of aperiodic arrivals are given in brackets. In the following text, we will refer to the clusters using a short-hand notation consisting of the

Table 5.5: Ship size cluster interval end c_i in meters, the total number a_i of calls in the clusters and number of aperiodic calls.

port	Gdańsk	Long Beach	Los Angeles	Le Havre
c_1	137 (115,5)	188 (89,1)	224 (167,9)	140 (216,6)
c_2	151 (103,2)	232 (137,2)	261 (132,4)	210 (337,25)
c_3	183 (121,1)	273 (118,11)	279 (180,5)	245 (333,7)
c_4	210 (11,3)	302 (203,14)	295 (252,12)	278 (400,19)
c_5	300 (17,10)	338 (254,15)	305 (181,12)	300 (355,29)
c_6	368 (50,2)	368 (193,10)	335 (294,13)	368 (528,49)
c_7	399 (54,1)	399 (1,1)	399 (102,16)	399 (104,7)
port	Hamburg	Rotterdam	Shanghai	Singapore
c_1	141 (905,8)	102 (153,2)	101 (642,2)	172 (4552,21)
c_2	170 (956,6)	141 (2395,13)	148 (4144,11)	198 (3206,21)
c_3	213 (298,5)	152 (532,2)	183 (2263,20)	225 (2231,21)
c_4	279 (302,23)	170 (444,4)	237 (1584,30)	262 (2182,47)
c_5	338 (339,45)	223 (240,26)	297 (1800,65)	302 (3188,67)
c_6	369 (422,27)	273 (154,13)	348 (1022,53)	345 (1635,28)
c_7	400 (72,6)	305 (80,11)	367 (151,14)	400 (1500,18)

initials of the port name and the cluster number starting from 1 for the shortest ships. For example, RT3 is the third cluster for Rotterdam, with 532 calls including 2 aperiodic arrivals (cf. Tab.5.5).

Discussion. It can be seen in Tab.5.5 that size classes are distinctive for each port. However, some similar clusters can be identified among large ships: {GD7, LB7, LH7, HB7}, {GD6, LH6}, {LB5, LA6}, {SH6, SI6}, {LB6, HB6}. The clustering obtained has its peculiarities. For example, there is only one ship in LB7 and she is aperiodic. This cluster will require a special handling: the ship data can be used directly as a representative for the LB7 cluster.

We built different size ranges $(c_{i-1}, c_i]$ for each port. Alternatively, the same ranges can be defined for all ports in the world. We will present ship sizes from this perspective in Section 5.5 (cf. Fig.5.10). An advantage of this alternative is that it makes port comparisons easier. A drawback is that some of such classes may be empty in some ports and the number of classes must be bigger to sufficiently discern ship size differences between ports, which complicates the STM. We decided to use size ranges specific to each port to get smaller numbers of classes and simpler STMs.

5.3.3 Ship Length Model

The easiest way of representing ship lengths in a certain class is to unify the whole cluster and to use the upper end of the cluster range. That is, model $\mathcal{L}_i$ for a cluster with range $(c_{i-1}, c_i]$ is c_i. Let us call this representation the longest-ship model.

Discussion. A more precise analysis of ship lengths in clusters reveals data and clustering peculiarities. In Fig.5.2 empirical cumulative distribution functions (CDF) of the ship lengths in example clusters are shown against theoretical CDFs of selected continuous distributions fitting ship lengths best. Parameters of the probability distributions chosen by the `fitdistrplus` method of R language are available from [89]. Fig.5.2 shows that there are clusters which follow very well the above longest-ship approach. For example, the ship lengths in cluster LH7 come from only five lengths in range $[395, 399]$m (Fig.5.2a) and rounding them up to 399m is not a substantial loss of precision. There are also clusters which would be better split into a few sub-clusters. For instance, visually LB6 (Fig.5.2b) could be much better divided into clusters of lengths $\approx$ 350m and $\approx$ 365m. Similar results were obtained for LA7 (not shown here). Thus, a better ship length model would be possible if more size classes were used for particular ports. However, this leads to a dilemma whether to emulate particular ports in more detailed way (at the cost of more complex model), or quite contrarily, generalize the results. Since our goals are in generalizing patterns in ship traffic, we do not follow the first path. Finally, there are clusters which ship lengths can be quite well approximated by continuous distributions. This is the case for SI1 and SH2 (Fig.5.2c, d). However, the reader should be aware that for some clusters (e.g. as narrow as LH7) searching for a continuous ship length distribution is unfounded.

5.3.4 Processing Time Model

Our first attempt to develop a vessel processing time model consisted in calculating a linear regression function of the processing time in the ship length: $p_j = a_1 L_j + a_2$, for all the ships of the port considered. In Fig. 5.3 examples of ship processing times p_j and processing time per unit of ship length p_j/L_j are shown. In all figures, the linear regression line, its parameters, and the coefficients of determination R^2 are also given. It can be observed that the distributions of p_j and p_j/L_j depend on the ship size. Usually, longer ships have larger p_j, but a negative correlation between p_j/L_j and L_j can be observed. However, it is obvious from Fig. 5.3 that

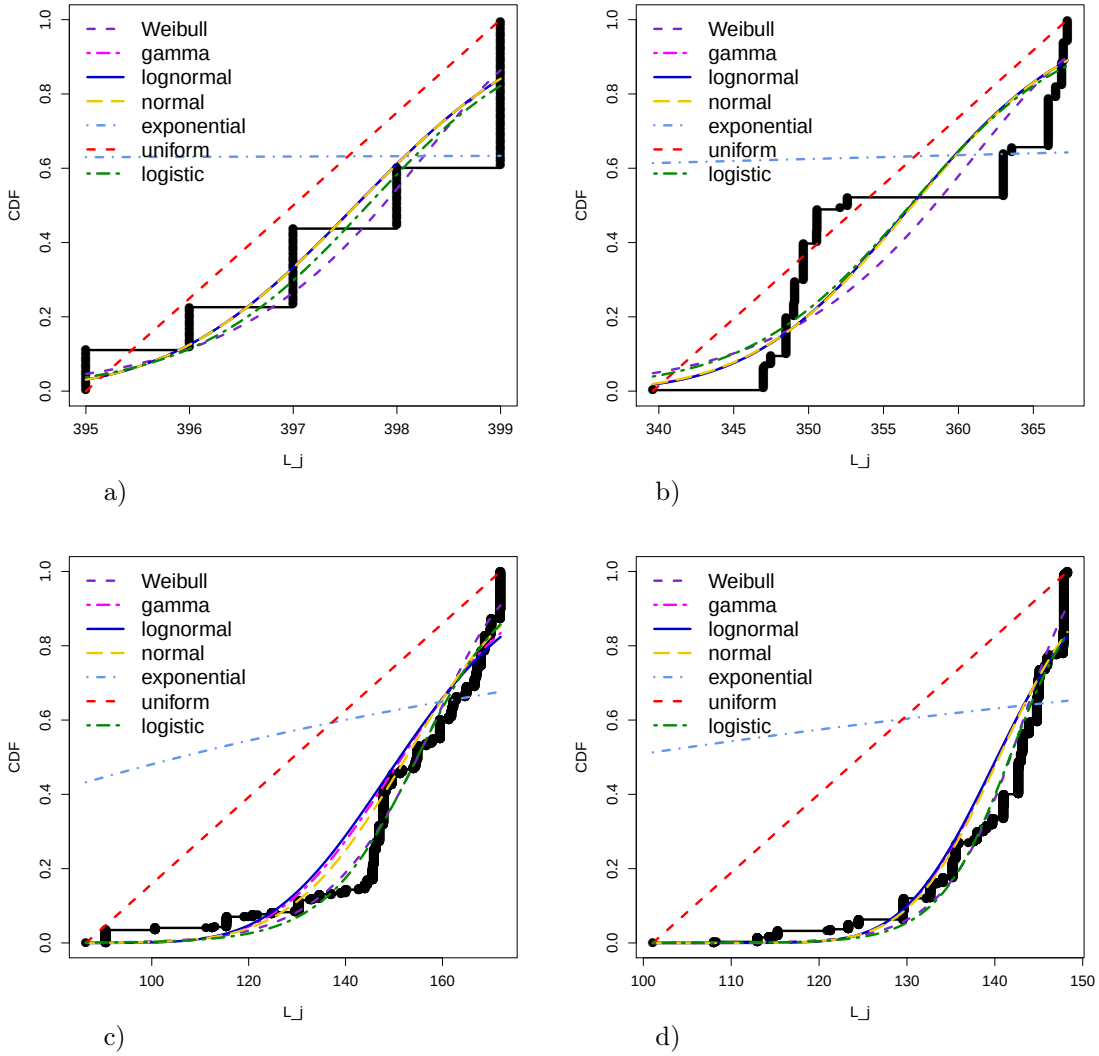


Figure 5.2: Empirical (bold black) vs theoretical ship length distributions in example clusters. a) LH7, b) LB6, c) SI1, d) SH2.

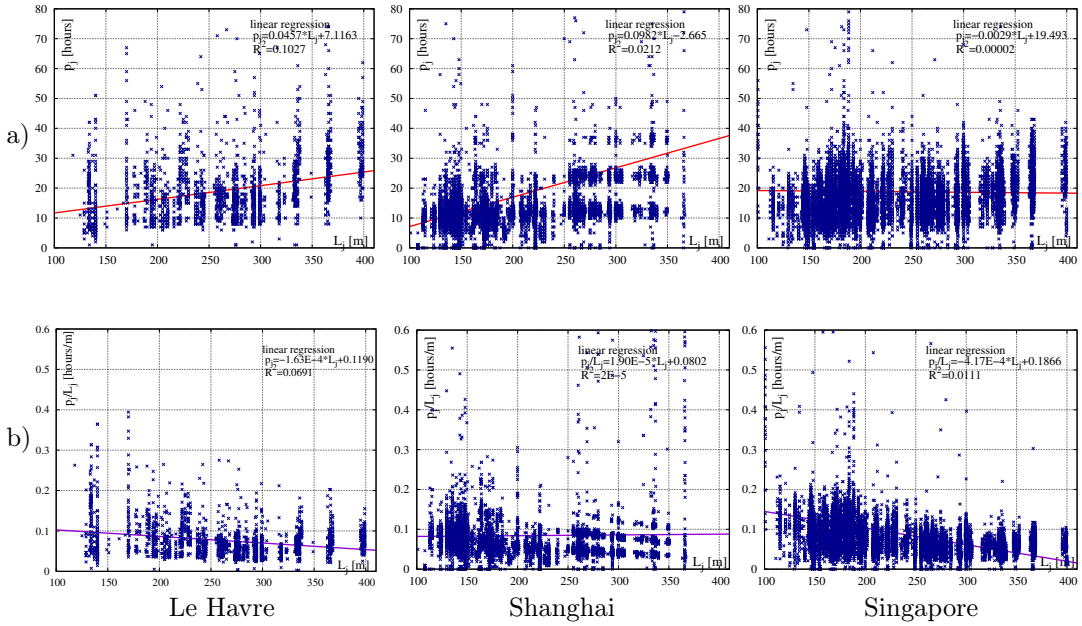


Figure 5.3: Examples of relationships a) p_j vs L_j (restricted to $p_j \leq 80$), b) p_j/L_j vs L_j (restricted to $p_j/L_j \leq 0.6$). Le Havre (left), Shanghai (center), Singapore (right). Linear regression fit and coefficients of determination are: Le Havre: a) $p_j = 0.0457L_j + 7.1663, R^2 = 0.1027$, b) $p_j/L_j = -0.000163L_j + 0.1190, R^2 = 0.0691$; Shanghai: a) $p_j = 0.0982L_j - 2.655, R^2 = 0.0212$, b) $p_j/L_j = 0.000019L_j + 0.0802, R^2 = 2E - 5$; Singapore: a) $p_j = -0.0029L_j + 19.493, R^2 = 2E - 5$, b) $p_j/L_j = -0.000417L_j + 0.1866, R^2 = 0.0111$. p_j in hours, L_j in m.

linear regression is not a good predictor of p_j. There is a great deal of dispersion both in p_j and in p_j/L_j. This is reflected in very low coefficients of determination R^2. Thus, linear regression is not suitable to reproduce accurately the processing times. It can be seen that p_j values align vertically in a way corresponding with ship size classes or particular ships. Hence, it is advisable to analyze ship processing times for distinct ship size classes, rather than using a single model for all possible lengths. In the following, we decided to consider p_j/L_j rather than p_j distributions because the former are more compact and their ranges are more similar between the ports. Furthermore, p_j/L_j can be considered as a better score for the logistic technologies used in the port than p_j. Correlations between L_j and p_j/L_j can indicate the economy of scale effect for the port considered. This will be discussed in Section 5.5.

Eight common parametric probability distributions: beta, exponential, gamma, normal, log-

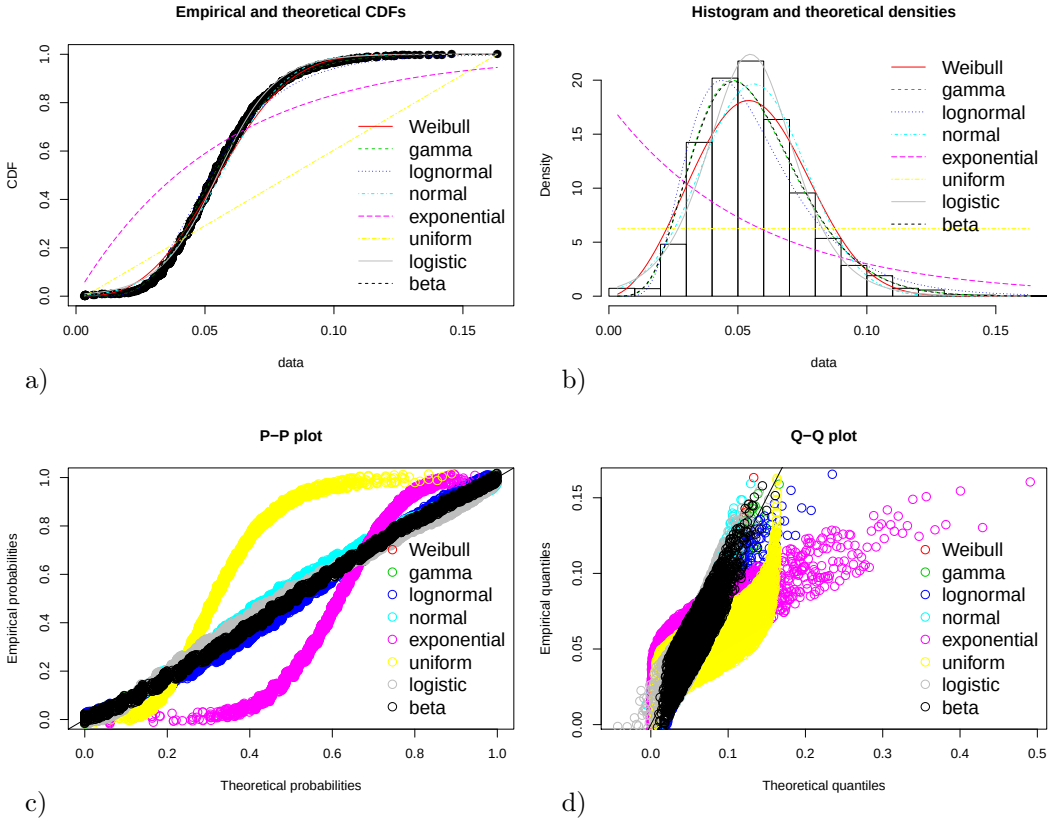


Figure 5.4: Example of visual output from `fitdistrplus` for SI5: a) cumulative distribution functions, black points are the actual observations, b) probability density functions, c) PP plots, d) QQ plots.

normal, logistic, uniform, Weibull, were fit to the p_j/L_j values for ship size clusters of the ports considered, by using the R language `fitdistrplus` package [20]. Distribution parameters were calculated using maximum likelihood estimation. Examples of `fitdistrplus` package visual output are shown in Fig.5.4. Visual data analyses are often unrestrictive, while numeric goodness-of-fit statistics often disagree in their recommendations. Moreover, we operate on a dataset representing a real ship stream which is rich in information and peculiarities. To make the STM building process deterministic and reproducible, we used Anderson-Darling goodness-of-fit score provided by `gofstat` from R programming language to choose the best fitting probability distribution for each port and ship size class. The distributions selected are summarized in Tab.5.6. The parameters of the best fitting distributions are provided in Appendix D.3 and in [89].

Discussion. For each distribution, the number of size clusters that are the best fit (wins), the number of clusters that are the worst fit (worst), and the best fit clusters, are given in Tab.5.6. The best fit clusters are the ones for which the distribution considered had the best fit according to Anderson-Darling goodness-of-fit score. Analogously, the number of worst fits is the number of cases for which the distribution considered provided the worst fit. The maximum number of possible wins is 55, i.e., the number of all ship clusters in all ports but LB7 which has exactly one aperiodic ship. Let us note that uniform distribution was excluded from Tab.5.6 because it is always the worst choice. Exponential and normal distributions are the second and third worst after the uniform. Uniform, exponential, normal distributions very often used in simulation and analytical modeling perform bad here, which is an interesting qualitative observation for modeling scheduling problems or testing algorithm performance. Although the lognormal and the logistic probability distributions fit best the p_j/L_j distributions in many cases, none of the tested distributions is predominantly the best. This lack of regularity between ports, of p_j/L_j winning distributions, demonstrates that each port is unique when the processing time of the ships is considered.

Table 5.6: p_j/L_j distributions selected for the ship size classes and ports

distribution	No.of wins	No.of worst	Selected (winning) clusters
beta	5	0	GD4,RT5,LA4,SI5, SI6
exponential	1	49	SH1
gamma	10	0	GD3, LH6, LH7, RT1, RT3, RT4, LA6, SI1, SI3, SI4
normal	2	4	HB2, LA2
lognormal	18	2	GD1, GD5, GD6,LH1, LH2, LH3, LH5, HB3, HB4, HB6, RT2, RT6, LA1, LB4, SH5, SH6, SH7, SI2
logistic	14	0	LH4, HB5, RT7, LA3, LA5, LA7, LB1, LB3, LB5, LB6,SH2, SH3, SH4, SI7
Weibull	5	0	GD2, GD7, HB1, HB7, LB2

5.3.5 Ready Time Model

Since most of the arrivals are periodic (cf. Tab.5.3), we start with the returning ships. The non-returning ships will be dealt with in the following subsection.

Returning Ships

In Fig.5.5 examples of ship return times ρ_j in days are shown for Le Havre, Los Angeles, and Singapore. In Fig.5.5a return times ρ_j are presented vs ship lengths L_j, and in Fig.5.5b histograms of the return times are given. The most frequent ship return intervals can be identified as week multiplicities in Fig.5.5b, which is typical of shipping network design practices. It is expected that return times depend on the ship size class. Usually, the longer the ship, the longer the return times (Fig.5.5a). This is confirmed by positive correlation between L_j and ρ_j values. The small values of R^2, and the observed large variation of the return times, suggest that other parameters are involved, e.g., shipping network timetables, weather conditions or ship processing time variability. Hence, a model assuming some fixed return period would not fit well the observed large scattering of the return times.

In order to construct model $\mathcal{R}_i$ of returns in cluster i, we applied method `normalmixEM` from R package `mixtools` [6], to fit a *mixture* of normal distributions into return times of each ship size class for each port. Assume a set of ship return observations $\overline{\rho_i} = [\rho_1, \ldots, \rho_{a_i}]$ for cluster i is given, where a_i is the number of returns in cluster i. The method `normalmixEM` fits a probability

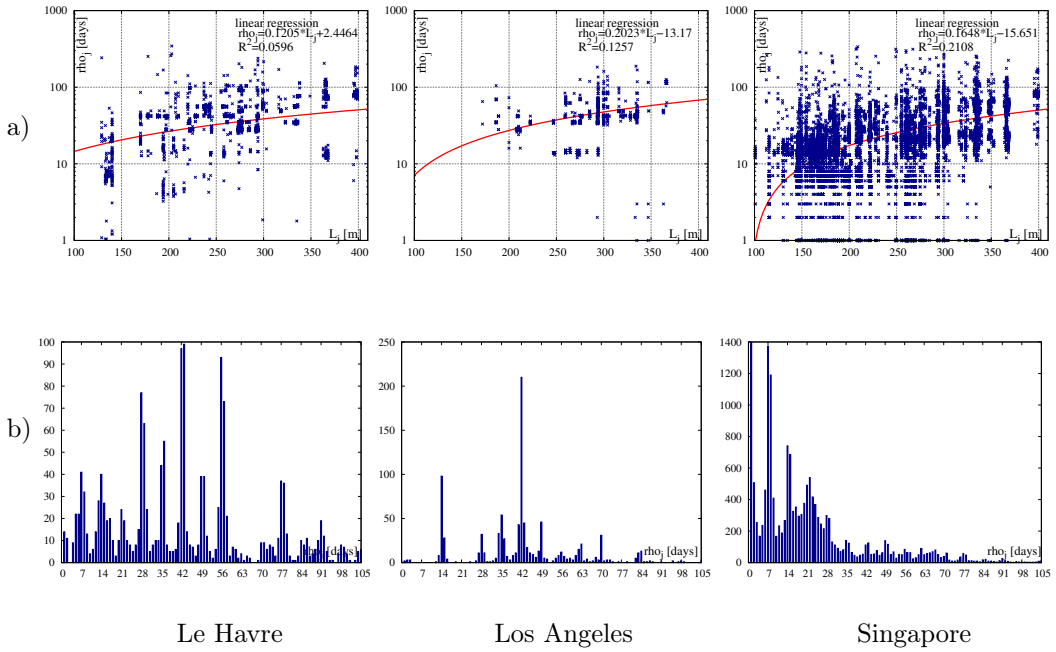


Figure 5.5: Examples of return intervals for Le Havre (left), Los Angeles (center), Singapore (right). a) Return intervals vs L_j, linear regression and coefficients of determination are shown in the pictures (vertical axis is logarithmic so linear function is not a straight line here), b) histogram of return intervals. ρ linear regression equations and coefficients of determination are: Le Havre: $\rho_j = 0.1205L_j + 2.4464, R^2 = 0.0596$; Los Angeles: $\rho_j = 0.2023L_j - 13.17, R^2 = 0.1257$; Singapore: $\rho_j = 0.1648L_j - 15,651, R^2 = 0.2108$. ρ_j in days, L_j in meters.

density function g which for some return value ρ_j can be written:

$$g(\rho_j|\overline{\theta}) = \sum_{h=1}^{\ell} \lambda_h \phi_h(\rho_j|\mu_h, \sigma_h^2), \tag{5.3}$$

where ℓ is the number of components in the mixture, ϕ_h is the normal probability density function with mean μ_h and variance σ_h^2, the λ_h are mixing proportions which are positive with a sum equal to 1, $\overline{\theta}$ is a density mixture parameter vector comprising ℓ triplets $(\lambda_h, \mu_h, \sigma_h^2)$. Mixture parameters $\overline{\theta}^*$ are chosen to maximize the fitting quality which is the logarithm of likelihood (*loglik* for short) of the data obtained:

$$\overline{\theta}^* = \underset{\overline{\theta}}{\operatorname{argmax}} \sum_{j=1}^{a_i} \log g(\rho_j|\overline{\theta}). \tag{5.4}$$

The obtained mixture parameter vector $\overline{\theta}^*$ is rather a local than a guaranteed global optimum. The run parameters of `normalmixEM` were set to default values, except for `maxit`=50000, `maxrestarts`=200 which were chosen experimentally to obtain results for the widest set of ship size classes and component numbers ℓ. Construction of mixtures with $\ell = 2, \ldots, 20$ were attempted. The number of components ℓ which provided maximum loglik in the verified range of values was chosen as the best mixture. Examples of visual results for fitting return times with mixtures (5.3) are shown in Figs 5.6, 5.7. The summary of return time models $\mathcal{R}_i$ for all clusters i can be found in Appendix D.4 and in [89].

Discussion. It is known that finding a matching mixture can be challenging [6]. Feasibility of `normalmixEM` depends on, e.g., size a_j of the data sample, number of mixture components ℓ and actual dispersion of the data. And indeed, mixtures could not be obtained for all ports, clusters and ℓ values. In particular for Gdańsk the range of feasible mixture components was the narrowest, often limited to just $\ell = 2, 3$. Moreover, there are only 4 different return periods in GD5. In most of the cases, the fitness quality (loglik) improves with the increasing number of mixture components ℓ. A large number of components in (5.3) is impractical so it is an attractive idea to limit the number of used mixture components. However, it is hard to choose a threshold of ℓ in an indisputable way. For example, in Fig.5.7a changes of loglik relative to the best obtained value (at $\ell = 20$) and density functions for $\ell = 11, 16, 20$, are shown for SI7. Though all shown density functions (Fig.5.7b-d) at least visually cover the data well, the range

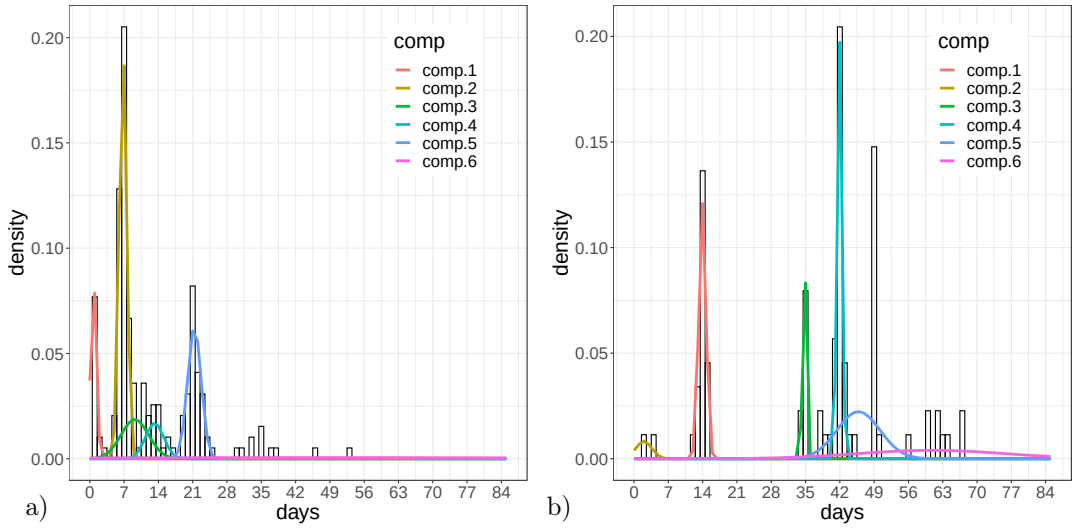


Figure 5.6: Examples of return times mixtures for a) LH1, b) LB3. Horizontal axes in days.

of loglik minimization is below 10%, but the value of loglik decreases slowly with ℓ (Fig.5.7a) and it is hard to point a single incontrovertible value of ℓ at which the process of increasing component number could be stopped. For reproducibility we stuck to the choice of $\ell \leq 20$ for which the loglik was the smallest.

On the basis of the identified return intervals, qualitative observations can be made for terminal planning and scheduling. Our study confirms that large ships call ports according to schedules with week multiplicities, but shorter schedules also exist. It can be seen in Fig.5.5b, that 6, 7, 8, 11 weeks return times are quite common. Thus, one week planning horizons (common in operational BAP) are adaptations that hardly represent a more complex process. In order to grasp interactions between different return periods, the least common multiple of the periods should be considered as a planning horizon, which easily exceeds a year.

We provided a probabilistic model of return times $\mathcal{R}_i$ for cluster i of a given port. It remains to define the first arrival. Several approaches are possible: 1. draw at random from normal distribution with mean μ_k and variance σ_k^2 where $k = \text{argmax}_\ell\{\lambda_\ell\}$; 2. draw returns many times from model (5.3) as if returning over many years to accumulate dispersion and use the offset from the beginning of the current year; 3. draw random day of week (DoW) and then draw random hour of day (HoD) using arrival distributions from Appendix D; 4. combine methods

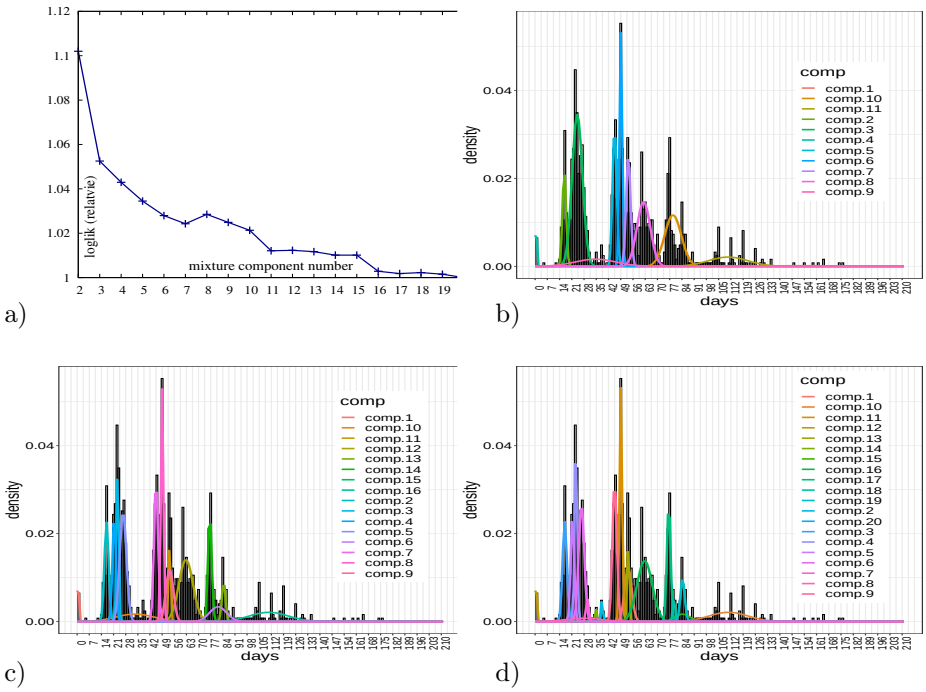


Figure 5.7: Examples of loglik changes with mixture component number ℓ in SI7 a) loglik vs ℓ, Density functions for b) $\ell = 11$, c) $\ell = 16$, d) $\ell = 20$.

by first using 1. and then moving the first arrival to the nearest day and hour generated by 3.

Non-returning Ships

Though the majority of the calls at ports are returning, aperiodic arrivals are not exceptions and there are over 50 ships in most of the analyzed ports, which is more than one arrival weekly. Hence, we decided to analyze this group of ships as a potential component of the STM. The number of aperiodic ships in each size cluster was given in Tab.5.5. Aperiodic arrivals are not concentrated in a restricted subset of clusters. Hence, ship classes and lengths can be modeled in the same way for all ships, both returning and non-returning. As the processing time model $\mathcal{P}_i$ for non-returning ships, we propose to use the model of the corresponding size class (Section 5.3.4). Note that even for the clusters with the biggest number of aperiodic arrivals (see Tab.5.5, SH6: 53, SH5: 65, SI5: 67) on average there is slightly more than one aperiodic ship per week in a size class. The median number of aperiodic ships per week in one size class calculated over all ports and clusters is 0.211 which is roughly equivalent to one aperiodic arrival per cluster

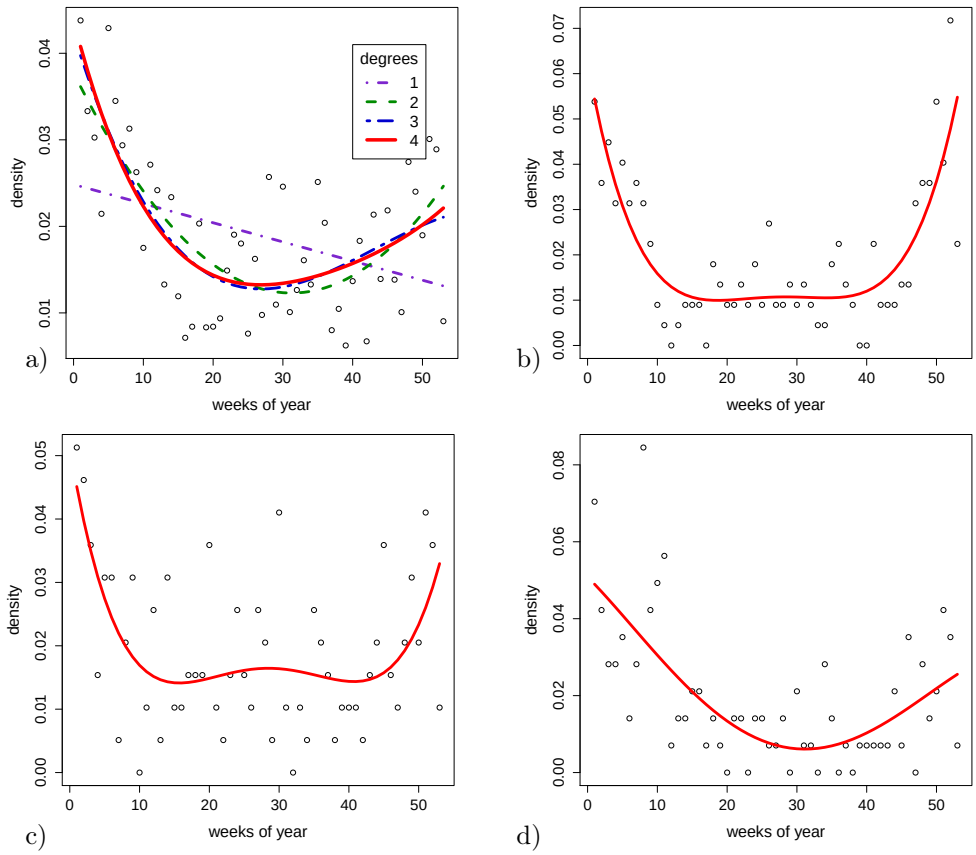


Figure 5.8: Examples of fitting polynomials for aperiodic arrivals over year's weeks. a) Fitting various degree polynomials into the normalized sum of frequencies in all ports. The weekly frequencies and the best-fit 4th-degree polynomial for b) Singapore, c) Shanghai, d) Le Havre.

over 33 days (≈ 4.7 weeks). Hence, aperiodic arrivals are not common enough to build *separate* and trustworthy statistical models of ready times for each size class with high time resolution. Therefore, to model the non-returning ship arrival process with a similar resolution as for the returning ships we will apply a superposition of three distributions: for the week of a year, for the day of a week DoW, and for the hour of a day HoD. This was done for whole ports, without distinguishing size classes.

Weeks of the year. It appeared that the number of weekly aperiodic arrivals changes over the year and in most of the cases the number of aperiodic arrivals increases at the turns of the

year (see Fig.5.8a). Thus, aperiodic arrivals show seasonal fluctuations. In order to represent the variability over the weeks of the year we decided to build an equivalent of a density function by fitting a polynomial $\eta(w)$ of week number w into the weekly frequencies. A careful study led to use polynomials of 4-th degree, which is the lowest even degree allowing a smooth transition between consecutive years. Visual output of the fitting can be verified in Fig.5.8a. Then specific 4th degree polynomials were obtained by fitting the data for each port. Examples of visual output can be seen in Fig.5.8b-d. The coefficients of the polynomials fitting the relative frequencies of aperiodic arrivals are given in Appendix D.6.1.

Days of a week and hours of a day. Examples of the distributions of aperiodic arrivals over days of a week for Le Havre and Rotterdam are shown in Fig.5.9a. Along the vertical axis fractions of the total number of aperiodic arrivals are shown. In Fig.5.9b the coefficient of variation for aperiodic arrivals over days of week is shown vs the number of aperiodic arrivals in the port during the year. Results for Le Havre are shown in Fig.5.9a because they present an appealing A-shaped pattern with the top arrivals on Wednesdays. In Rotterdam, Fridays were days with $2.15 \times \sigma$ departure from the average, where σ is the standard deviation of the daily fractions in the port. This was the biggest departure from average in all studied ports. It may be the case that these two ports exhibited some tendency toward processing non-returning ships on these two days. However, it may be also a human perception artifact. As many as 64% of aperiodic arrivals of all ports on days of a week fit in $1\times\sigma$ range around the average. The exceptional "Rotterdam Friday" may emerge randomly with probability 0.019 if dispersion of aperiodic daily arrivals is normally distributed. Moreover, it can be seen in Fig.5.9b that ports with bigger number of aperiodic ships have smaller dispersion of arrivals between days of a week. Hence, there are good reasons to think that the number of aperiodic arrivals changes randomly over the days of a week. Therefore, we applied `fitdistrplus` method from R programming environment to find the distribution fitting best the variability of fractions of aperiodic arrivals over days of a week. For example, the best fit of aperiodic arrivals over days of a week aggregated over all ports is the logistic distribution. Similar analysis was conducted for aperiodic arrivals over hours of a day. Results are collected in Appendix D.6.2 and D.6.3.

Discussion. We proposed a 4th degree polynomial of the week number in a year as ready time model. This function can be used as an empirically-built probability density function of aperiodic arrivals in a given week. By applying a continuous function we attempted to extract a smoothed

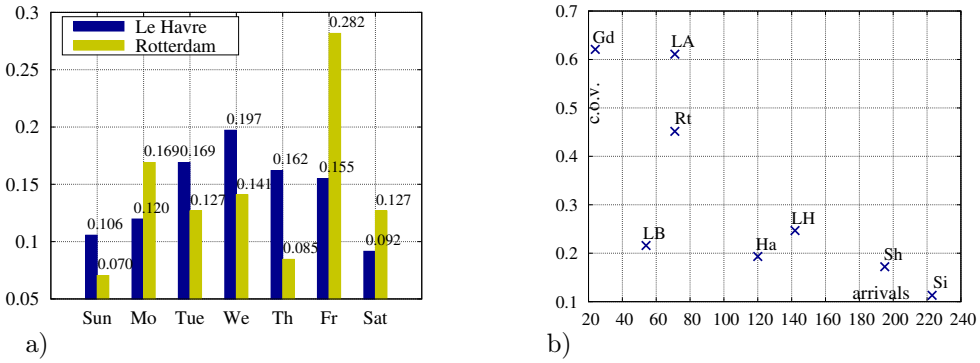


Figure 5.9: Examples of aperiodic arrivals over days of a week. a) Fractions of arrivals for Le Havre and Rotterdam. b) Coefficient of variation vs number of aperiodic arrivals per year for different ports.

pattern in the central tendencies of weekly arrivals. However, an aperiodic arrival distribution can be defined in alternative ways. For example, as a list of probabilities for particular weeks averaged over ports. An advantage of such a representation is simplicity. A disadvantage is the lack of smoothing and generalizing effect of a continuous function. The lack of data for multiple years impedes constructing more reliable models both continuous and time-discretized.

The 4th degree of the fitting polynomial is minimum which allows smooth transition between years, which seems a practical requirement. But for reasons of computational simplicity lower degree polynomials may also be acceptable. A visual comparison of the 2nd and 3rd degree polynomials in Fig.5.8a suggest that the advantage of fitting as large as a 4th degree polynomial is minor.

Due to shortage of data, quality of models for aperiodic arrivals over days of a week (DoW) and hours of a day (HoD) is low. Hence, for the reason of confidence in the models it seems more legitimate to use analogous models for all ships, i.e. both periodic and aperiodic. It may be also considered advantageous because with respect to assigning service time in a day of the week and hour of the day, aperiodic ships are handled in the same way as periodic ships.

Let us conclude about arrival time modeling. In the ship arrival time models, various decision levels intersect. On the one hand, it was possible to build a continuous model for periodic arrivals. This model incorporates and generalizes traffic patterns resulting from, e.g., global network of main shipping lines or delays due to weather conditions. On the other hand, due to

Algorithm 1 Instance generator(STM, N, H)

```
1: j ← 1;
2: for cl in 1 to 7 do
3:    a_cl ← N × a_cl^STM / Σ_{k=1}^{7} a_k^STM;
4:    while a_cl > 0 do
5:       choose length L_j of the ship in call j as described in Section 5.3.3;
6:       if Rnd(1)≤ η_a then
7:          ArrNum ←DrawAperiodic(j, cl, L_j)
8:       else
9:          ArrNum ←DrawReturning(j, cl, L_j, a_cl)
10:      end if
11:      j ← j + ArrNum;
12:      a_cl ← a_cl − ArrNum;
13:   end while
14: end for
```

lack of data and need for representing aperiodic arrivals with the resolution comparable to the periodic arrivals we resorted to use DoW and HoD models. Yet, the weekly and daily operations of ports are subject of online optimization these days, and hence, the DoW and HoD models may change and become deterministic by nature.

5.4 Applying the Model

In this section we describe how to apply our STM to generate instances of the ship traffic with the above features. The steps to be taken are outlined in Algorithm 1. It accepts ship traffic model STM, the number of calls N, and the interval H of arrivals as input parameters. Particular ports are emulated by the input STM. An example of STM for cluster LH1, com-

Table 5.7: STM for Le Havre ship size class 1 (LH1, the shortest ships)

0.0951	a_1 – fraction of class 1 ships in the whole set of container ships
[118,140]	range of ship length in class 1 [meters]
lognormal	type of p_j^s/L_j^s distribution [hours per meter]
-2.44661	parameter of lognormal distribution, expected value of $\ln(p_j^s/L_j^s)$
0.68494	parameter of lognormal distribution, standard deviation of $\ln(p_j^s/L_j^s)$
6	number of components in the Gaussian mix representing ship return times, the components are defined by mixing proportion, mean value [days], standard deviation [days]. The components are given below:
(0.1151, 4.6301, 0.8343) (0.1740, 15.5915, 2.0145) (0.0685, 21.0844, 1.0550) (0.2201, 36.8575, 22.4692) (0.3928, 42.0277, 0.7392) (0.0296, 201.0784, 89.3193)	

Algorithm 2 DrawAperiodic(j, c, L_j)

1: generate number x according to the continuous distribution of ratios p_c/L_c for the cluster c (Section 5.3.4), set processing time $p_j = x \times L_j$;
2: choose arrival date r_j according to the model in Section 5.3.5;
3: record (j, L_j, p_j, r_j)
4: return 1;

Algorithm 3 DrawReturning(j, c, L_j, a_{max})

1: generate first arrival *date* as proposed in Section 5.3.5; $count \leftarrow 0$;
2: **while** $date < H$ and $count < a_{max}$ **do**
3: generate number x according to the continuous distribution of ratios p_c/L_c for the cluster c (Section 5.3.4) set processing time $p = x \times L_j$ for call $(j + count)$;
4: record$(j + count, L_j, p, date)$;
5: $count \leftarrow count + 1$;
6: choose return time ρ according to mixture of distributions (Section 5.3.5);
7: $date \leftarrow date + \rho$;
8: **end while**
9: return *count*;

prising the shortest ships in Le Havre is shown in Tab.5.7. The intensity of ship traffic $\iota = N/H$, measured by the number of arrivals in some time interval, is regulated by the number of calls N and the interval H of arrivals. The size classes are generated in loop 2-14. The number of arrivals a_{cl} for cluster cl is calculated using the numbers of arrivals a_{cl}^{STM} in the STM built on historical data. Arrivals are generated in loop 4-14 and j is the total arrival counter. The size of the arriving ship L_j is chosen in step 5. The modeler has to make a design decision here on the ship length L_j generation method, according to the options outlined in Section 5.3.3. The type of arrival is chosen in step 6: it is an aperiodic arrival with probability η_a, otherwise it is a returning ship. Rnd(1) is a pseudo-random number generator providing numbers in range [0,1] with uniform distribution. Function DrawAperiodic generates data for an aperiodic arrival in cluster cl of a ship with length L_j. Function DrawReturning generates at most a_{cl} arrivals of the returning ship in cluster cl of length L_j as sequences of arrivals. The two functions return the number of generated arrivals stored in $ArrNum$. The global arrival counter j is increased and the remaining number of arrivals a_{cl} is decreased in steps 11 and 12, respectively.

Function DrawAperiodic is shown as Algorithm 2. In steps 1 and 2, processing time and arrival date are generated. In step 3, a tuple of values defining jth arrival is recorded. Function DrawReturning is defined as Algorithm 3. In DrawReturning a sequence of the same ship arrivals

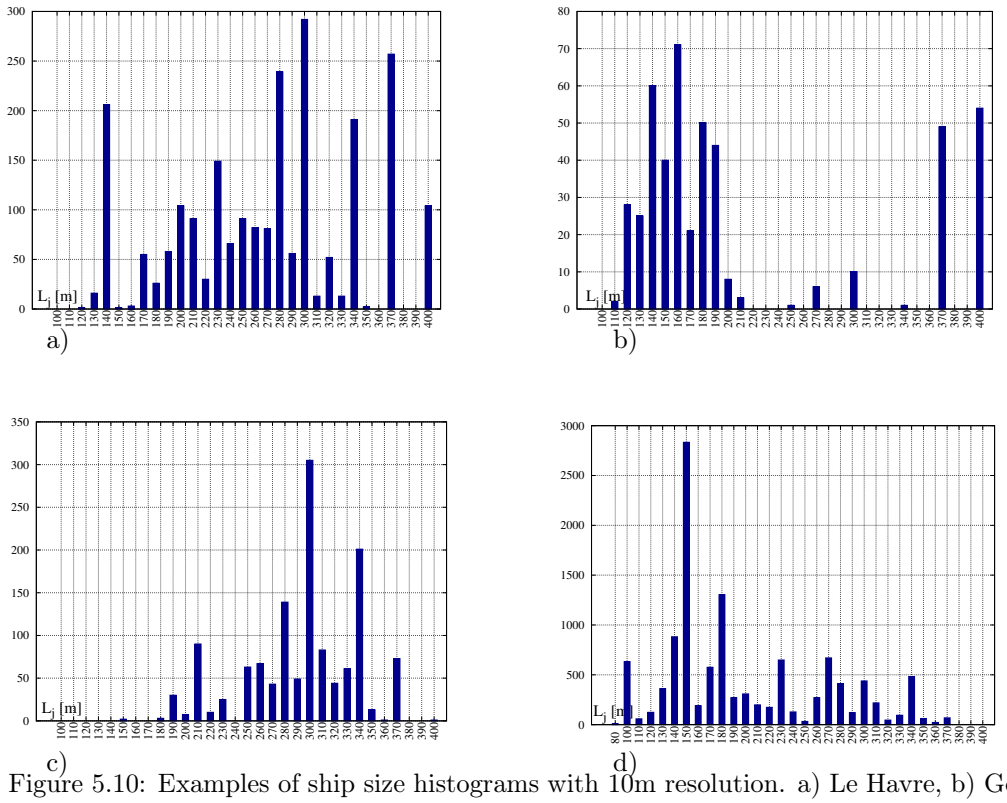


Figure 5.10: Examples of ship size histograms with 10m resolution. a) Le Havre, b) Gdańsk, c) Los Angeles, d) Shanghai.

is generated in loop 2-8. The loop execution stops if the next arrival is after the end of time horizon H or sufficient number of arrivals is created. In line 7 the next arrival time is calculated.

5.5 Distinctive Port Features

As shown in the previous sections ports differ in many ways. The specific features that appear in our data can help to choose the right model for some other port.

One expected port dissimilarity is in the mixture of ship sizes. Different length patterns are expected due to the characteristic location, like the existence of a chain of neighboring ports, or the location in the river estuary. Examples of ship size histograms covering all size classes with the same box ranges are shown in Fig.5.10. For example, Los Angeles (Fig.5.10c) has no small container ships because there are no river waterways nearby and even local traffic is oceanic.

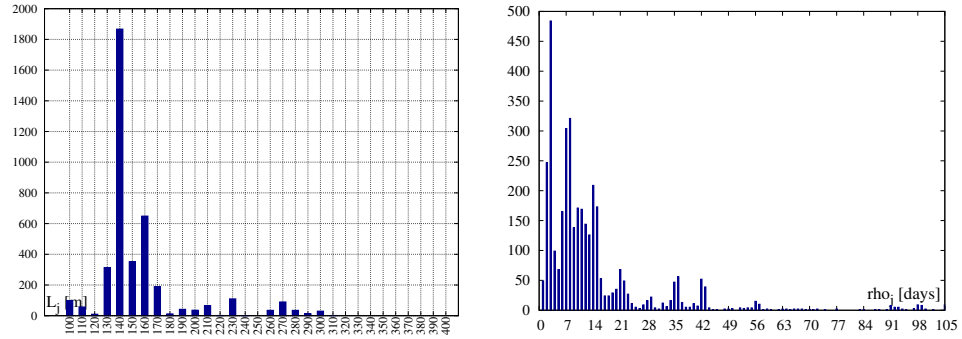


Figure 5.11: Rotterdam a) L_j and b) return period ρ_j histograms.

Similar pattern of ship sizes can be found in Long Beach (not shown here). Conversely, Gdańsk and Shanghai (Fig.5.10b, d) are in a chain of local ports, and what is more, Shanghai port is in large rivers estuaries. Hence, there are large fractions of small ships in the traffic. Though Le Havre is in the Seine estuary it is also the westernmost ocean port in a chain of European ports and hence there is a significant fraction of medium size ([150,300]m) ships for local connections (Fig.5.10a). Similar pattern with the domination of medium size ships exist in Singapore (not shown here).

The pattern of ship sizes is connected with the return intervals because small ships operate locally, whereas the largest ones operate in the long ocean lines. This can be verified in Figs 5.5b and 5.10a,c for Le Havre, Los Angeles, as well as in Fig.5.11 for Rotterdam. In Le Havre there are a broad set of ship size classes and return periods because it is an ocean port, in a chain of local ports. There are almost no short ships and short return times in Los Angeles because it is an ocean port. In Rotterdam (see Fig.5.11) ships shorter than 180m and returning in less than 3 weeks dominate, and very large vessels are not present. This is understandable for an inland port on the banks of Nieuwe Maas (the Maasvlakte terminals traffic is not included into the analyzed data).

Let us conclude the above discussion with a recommendation on suitable STMs for optimization problems of port logistics. Location is the key selection determinant. An STM of the most similar port should be used. If the location considered for optimization is an isolated ocean port, then Los Angeles or Long Beach can be chosen. If the port is in a chain of local ports, choose

Table 5.8: Correlation between p_j/L_j and L_j.

port	Gdańsk	Long Beach	Los Angeles	Le Havre
r	-0.007	0.065	0.389	-0.263
SE_r	0.046	0.032	0.025	0.020
port	Hamburg	Rotterdam	Shanghai	Singapore
r	-0.273	-0.185	0.005	-0.105
SE_r	0.017	0.016	0.009	0.007

Le Havre or Singapore STMs. If, furthermore, the location is in big river system estuary, then Shanghai STM can be recommended. For an inland port, our Rotterdam STM can be applied.

Another interesting feature is the relationship between the ship length L_j and its processing time p_j. It can be expected that in general longer ships have longer processing times. In order to compensate for this rather obvious relationship we decided to investigate the correlation between L_j and p_j/L_j (see also Section 5.3.4). It seems intuitive, that such correlation either should not exist (to treat different size classes fairly) or may be negative (because terminal operators would rather dedicate more resources and effort to the largest ships to take advantage of the economy of scale). The coefficients of correlation r between L_j and p_j/L_j and the standard errors SE_r of these coefficients are shown in Tab.5.8. The correlations in Tab.5.8 are weak according to the frequently used rule of thumb requiring that $r \geq 0.7$ for a strong correlation. If we use the rule that range $[r - SE_r, r + SE_r]$ does not include 0, then it can be observed that there are ports where the correlation is nonexistent (Gdańsk, Shanghai) which can be interpreted as a fair treatment of the different classes. There are ports where the correlation is negative as expected. Surprisingly, there are also ports (Los Angeles) where longer ships do take longer time (disproportionately to L_j) to be served. A closer look at the terminal management policy is needed to explain these effects.

Let us finish this section by observing that with a growing number of arrivals N the coefficient of variation of the arrivals calculated over to days of a week and hours of a day decreases (Fig.5.9b, Fig.D.5) and the feasibility of fitting a continuous distribution as a model of L_js for size clusters improves (Figs 5.2c,d).

5.6 STM Summary

In this chapter, we studied container ship traffic patterns in eight ports of the world to develop traffic models for optimization and simulation of port logistics. These models will be used to study SQPP in Chapter 6. We developed more advanced and detailed models than existing previously in the literature. The models respect the relationships between ship sizes, processing times and arrival times. Each size class has its own processing time and arrival time statistical representation. Particular attention is paid to return times and their dispersion, but aperiodic ships are also considered.

Representing real-world phenomena in mathematical models is full of trade-offs. One of our goals was to generalize the observed traffic patterns, but generalizing leads to loss of information. For example, establishing some general model framework is possible, but particular ports require their own variants to adjust not only the distribution numerical parameters but also the actual distribution types (see Tab.5.6). Hence, a trade-off exists between accuracy when keeping details of a given port, and simplicity when using one model with a unique distribution.

Since we decided to build models representing each port separately, a decision-maker should choose between these ports and treat them as archetypes of ocean ports, ports in local port networks, or ports with local river traffic.

A realistic STM provides indications on realities of the assumptions often made when analyzing optimization problems in port logistics. Thus, our STMs allows to avoid ad hoc choices in researching such problems. We established that: 1) ship length is the key determinant for return and service times (hence these three parameters are strongly related), 2) return times have a strong periodic component (thus memoryless distributions *are unjustified*), 3) typical distributions (uniform, normal, exponential) are the least suitable to represent service times, 4) aperiodic arrivals have a strong seasonal component, 5) ship return time patterns determining time horizons of planning and scheduling easily span over years. 6) Although the mathematical structure of the model is the same for all ports, its instantiations differ. As a side benefit, this gives also a way to compare ports.

The ship traffic model introduced in this thesis is explainable. That is, the model not only generates ship stream with given characteristics, but it can also help to explain how and why these characteristics emerge. Thus, rather than some machine-learning "black-box" (cf. [31]),

we use data analysis and explicit statistical distributions. Thanks to this, the model parameters can be used directly in algorithms solving certain port logistic problems. Let us note that our ship traffic model is not predictive in the classic sense in which, e.g., time series analysis is predictive. We do not intend to predict the future number of TEUs or calling ships on the basis of some independent determinants. The model is built to take advantage of recreating features of the real traffic in the above-mentioned simulation applications. The number of calling ships is an input parameter. It is assumed that according to the current state of the art, numbers from determined probability distributions can be generated pseudo-randomly with satisfactory accuracy.

Chapter 6

Stochastic Quay Partitioning Problem

In this chapter the earlier research threads are joined. Let us outline the SQPP solution setting (cf. Fig.6.1) and the reasons behind it. A solution of an SQPP defines a *quay partition* which is a vector of berth lengths. The partition should be evaluated on the future ship traffic. Since it is unknown, SQPP solutions are evaluated on test scenarios simulating future ship stream. The scenarios are generated according to a ship traffic model (STM) built in Chapter 5. Evaluating a quay partition on some test scenario consists in scheduling ships on berths and assessing quality of service, e.g., ship waiting time. As argued in Chapter 2 to evaluate an SQPP solution it is inevitable to: 1) refer to solving BAP, 2) consider very large instances, 3) use specialized methods. We will use tailored algorithm portfolios considered in Chapter 4 to solve very large BAPs under a runtime limit. For each partition a population of scenarios and their evaluation scores are collected. The partition with the best central tendency score (e.g. mean) is considered the best SQPP solution.

6.1 Stochastic Quay Partitioning Problem Formulation

Although Quay Partitioning Problem for a single arrival scenario (i.e. DQPP) has already been defined in Section 3.2, we add here new elements to handle stochasticity. An instance of SQPP is

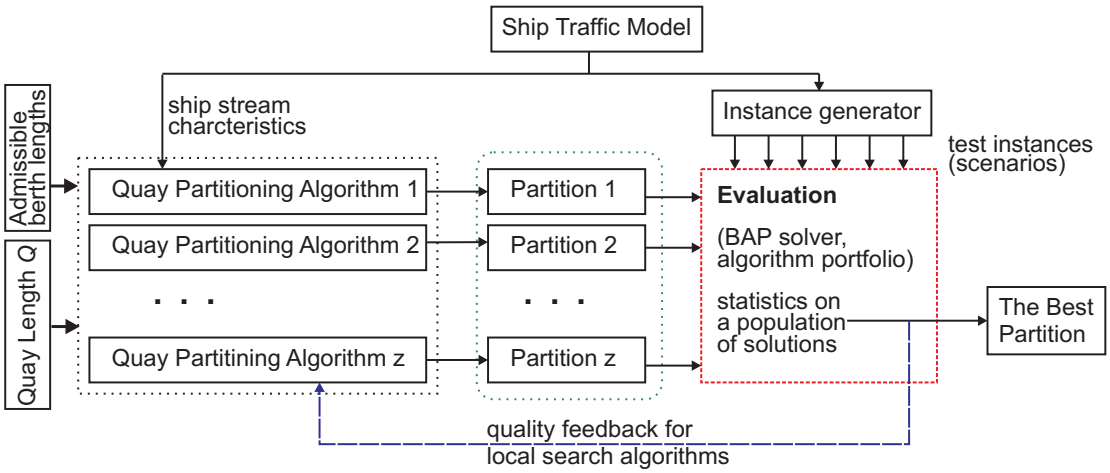


Figure 6.1: SQPP solution workflow

defined by: quay length Q, set $B = \{\lambda_1, \ldots, \lambda_f\}$ of admissible berth lengths, ship traffic model (STM) $\mathcal{S}$ with the parameters of ship arrival intensity ι and time hroizon H. Without loss of generality we assume $\lambda_1 < \cdots < \lambda_f$, and for simplicity of exposition, that Q is divisible by λ_1. $\mathcal{S}$ is a stochastic process generating ship traffic as scenarios. The STM models the fraction a_i of ship size class i in the total number of arrivals, ship arrival times, processing times and weights. Arrival intensity ι is an average frequency of vessel arrivals taken over the considered time horizon H. We will express ι in arrivals per week (a/w in short). For some scenario s, the number of ship arrivals $n_s \approx H\iota$ and for the arriving ship $j = 1, \ldots, n_s$, arrival time r_j^s, service time p_j^s, length L_j^s and weight w_j^s (profit from serving the ship) are determined. We assume that under any scenario there is an upper limit L_{max} on vessel lengths. Currently $L_{max} = 400$m is the size of the largest container mother ships.

A solution of SQPP is a partition of quay length Q into berths which can be represented as vector $K = (k_1, \ldots, k_f)$ of berth lengths $\lambda_1, \ldots, \lambda_f$ frequencies. By definition $Q \geq \sum_{i=1}^{f} \lambda_i k_i$ and $L_{max} = \max_{j=1}^{n_s}\{L_j^s\} \leq \max_{i=1}^{f}\{\lambda_i : k_i > 0\}$, i.e., for all ships sufficiently long berths exist. The total number of berths in some SQPP solution is $m = \sum_{i=1}^{f} k_i$. Vessels are positioned in the berths according to the 2in1 hybrid layout.

Let d denote the number of STM scenarios, c_j^{si} ship j service completion time under scenario s obtained by some scheduling algorithm i. A classic scheduling objective is mean weighted flow time (MWFT): $F(K, s, i) = \sum_{j=1}^{n_s} w_j^s (c_j^{si} - r_j^s) / \sum_{j=1}^{n_s} w_j^s$ for scenario s and scheduling algorithm

Table 6.1: Summary of notations for Chapter 6

a_i	fraction of ship size class i in the whole number of arrivals
B	set of admissible berth lengths
c_j^{si}	ship j service completion time for scenario s solved by algorithm i
d	number of scenarios
e	number of ship size classes in the ship traffic model
f	number of admissible berth lengths, $f = \|B\|$
$F(K, s, i)$	mean flow time for partition K, scenario s, scheduling algorithm i
$F(K)$	mean flow time for partition K averaged over all evaluations
g	number of algorithms in a portfolio
H	time horizon of the evaluation (simulation)
ι	weekly arrival intensity [a/w]
K	partition, vector of berth lengths frequencies
k_i	number of berth length λ_i occurrences in a partition
λ_i	admissible berth length i
L_i	upper end of ship length in size class i
L_j^s	length of the ship in the jth arrival in scenario s
L_{max}	maximum vessel length
m	number of berths in K, $m = \sum_{i=1}^{f} k_i$
n_s	number of vessel arrivals in scenario s
p_j^s	service time of the jth arrival in scenario s
Q	quay length
r_j^s	arrival (ready) time of the jth ship in scenario s
$\mathcal{S}$	ship traffic model (STM)
$\sigma(K)$	standard deviation in partition K evaluations
w_j^s	weight (value) of servicing ship j in scenario s

i. For d scenarios and a portfolio of g scheduling algorithms a population of $d \times g$ values of $F(K, s, i)$ is obtained. We will use the average of the MWFTs over the scenarios and algorithms:

$$F(K) = \frac{1}{dg} \sum_{s=1}^{d} \sum_{i=1}^{g} F(K, s, i) = \frac{1}{dg} \sum_{s=1}^{d} \sum_{i=1}^{g} \frac{\sum_{j=1}^{n_s} w_j^s (c_j^{si} - r_j^s)}{\sum_{j=1}^{n_s} w_j^s} \tag{6.1}$$

as the key quality criterion. We use average over algorithms (the sum over g) because it is a quality central tendency indicator, which is more reliable than, e.g., the results in one best algorithm. Let us observe that there may be further advantages of using algorithm portfolios to evaluate partitions. At the early strategic level of designing a container terminal, it is unknown what methods will be used by the terminal operators to assign vessels to the berths. A population of evaluations by various algorithms exposes the quay design process to a greater variety of possible scheduling methods, and thus, builds a broader view on potential dispersion of the partition scores under the future, yet currently unknown, BAP algorithms. Standard deviation

Table 6.2: Evaluation runtimes in seconds

intensity ι [a/w]	20	50	100	200	500	1000
Le Havre, $Q = 3500$m, runtime per scenario						
median	0.48	3.52	43.69	271.58	1814.19	6011.90
SIQR	0.18	2.50	16.88	48.56	220.44	296.65
Le Havre, $Q = 3500$m, runtime per scenario and algorithm						
median	0.06	0.44	5.46	33.95	226.77	751.49
SIQR	0.02	0.31	2.11	6.07	27.56	37.08
Shanghai, $Q = 5000$m, runtime per scenario						
median	0.92	2.92	7.68	31.89	1132.50	3973.68
SIQR	0.07	0.10	0.38	29.99	255.66	388.10
Shanghai, $Q = 5000$m, runtime per scenario and algorithm						
median	0.11	0.36	0.96	3.99	141.56	496.71
SIQR	0.01	0.01	0.05	3.75	31.96	48.51

SIQR – Semi-Inter-Quartile Range

$\sigma(K)$ in the population of values $F(K, s, i)$ will be the secondary criterion. For simplicity of the exposition indexes s, i will be dropped in the further text if particular scenarios are not singled out and the set of evaluating algorithms is constant.

6.2 Partition Evaluation

In order to evaluate quality of partition K on an STM-generated scenario s a BAP instance must be solved. Since SQPP emerges in long strategic port planning, arrival numbers n_s are large, cover portfolios (4.1)-(4.3) defined in Chapter 4 will be used. For the further study a portfolio of $g = 8$ algorithms constructed for runtime of at most 50s at $n_s = 10000$ was selected (compare Fig.4.5 in Section 4.7). The portfolio comprises heuristic algorithms: SPT-Prio, SPTGI-Prio, SPTGI-La2, GISPT-Prio, SAF-Prio, SAF-La2, RND-Prio, RND-La5. In the partition evaluations, scenarios for a particular STM were generated as described in Section 5.4, while ship weights were generated as $w_j^s = L_j^s p_j^s (0.5 + rnd(1))$, where $rnd(1)$ is pseudorandom number generator with uniform distribution in range [0,1]. The total number of vessels in a scenario is a product of one year time horizon and the given arrival intensity ι. Full model parameters and example scenarios can be found at [89].

In Tab.6.2 examples of evaluation runtimes for annual traffic with various weekly arrival intensities are given. For example, intensity 100 a/w is roughly equivalent to scheduling $n_s = 5200$ arrivals. In the tests the set of admissible berth length was $B = \{100, 200, 300, 400, 500, 600\}$m.

There were 10 scenarios for each STM (i.e. port) and intensity. All non-dominated partitions were tested (cf. Section 6.3.1, Section 6.4) to collect the data for Tab.6.2. Computational tests presented in this chapter were conducted in 2023 in Poznań Supercomputing and Networking Center, on Eagle/Altair cluster with Intel Xeon Platinum 8268 CPUs, each with 48 cores and 4GB RAM per core [85]. On the one hand, the computations were quite extensive. Over 43800 processor-hours were spent in the tests shown in Tab. 6.2. On the other hand, the runtimes per algorithm and per scenario are short, all evaluations of d scenarios by g algorithms are mutually independent and were run in parallel (see Section 6.6).

6.3 Partitioning Algorithms

In this section we introduce algorithms constructing partitions K for the given quay length Q, set of admissible berth lengths B and STM $\mathcal{S}$.

6.3.1 Brute-force Enumeration

The number of possible partitions grows exponentially with the number of berth sizes and can be upper-bounded by $O(Q^f / \prod_{i=1} \lambda_i)$. The actual number of interesting partitions is smaller because some partitions are dominated. Let $K = (k_1, \ldots, k_f)$ be a partition. If $\exists j : \lambda_j < Q - \sum_{i=1}^{f} k_i \lambda_i$ then K is dominated by $K' = (k_1, \ldots, k_j + 1, \ldots, k_f)$. Let us remind that there must be at least one berth of length greater or equal L_{max}. In Tab. 6.3 numbers of non-dominated partitions are given for some exemplary quay lengths and sets B of admissible berth lengths. The quay lengths in B were chosen from range [100,600]m with various resolutions: 200m, 100m, 50m, 25m. It can be concluded, that the number of non-dominated partitions is amenable to brute-force evaluation for contemporary quay lengths and small sets B (e.g. Q up to 5km and berth lengths with 100m resolution). In the following this method will be presented under short name BF.

6.3.2 Histogram Matching and Integer Programming

Suppose a histogram of ship lengths importance is given in an STM as a vector $(q_1, \ldots, q_e)$ for ship lengths $\{L_1, \ldots, L_e\}$, where $q_i \in [0, 1]$. The quay can be partitioned so that fractions of Q

Table 6.3: Number of nondominated partitions for exemplary ports

Berth lengths in [100m,600m] with $L_{max} = 400$m				
B with resolution	200m	100m	50m	25
Gdańsk $Q = 660$m	2	4	11	31
Le Havre $Q = 3.5$km	32	2052	187444	66871400
Shanghai $Q = 5$km	64	8958	2470285	4579673758
$Q = 10$km	233	188625	641225038	*
$Q = 26$km	1473	16738836	*	*

* – enumeration abandoned after 24h

dedicated to particular ship lengths match histogram $(q_1, \ldots, q_e)$ as closely as possible. Let δ denote the biggest difference between the required value q_j and the actual fraction of the quay length where ships of length L_j can be moored. Let u_{ij} be the number of length L_j vessels which can be positioned at berths of length λ_i. Admissible values of u_{ij} are: 0 for $\lambda_i < L_j$, 1 when $L_j \leq \lambda_i < 2L_j$ and 2 for $2L_j \leq \lambda_i$. Solution vector $K = (k_1, \ldots, k_f)$ approximating importance vector $(q_1, \ldots, q_e)$ can be obtained by solving the following integer linear program:

$$\textbf{ILP1}: \min \quad \delta \tag{6.2}$$

subject to:

$$\sum_{i=1}^{f} k_i \lambda_i = Q \tag{6.3}$$

$$\sum_{\{i:\lambda_i \geq L_{max}\}} k_i \geq 1 \tag{6.4}$$

$$\sum_{j=\ell}^{e} q_j - \sum_{j=\ell}^{e} \sum_{\{i:\lambda_i \geq L_j\}} u_{ij} k_i L_j / Q \leq \delta(e - \ell + 1) \qquad \text{for } \ell = 1, \ldots, e \tag{6.5}$$

$$k_i \in Z^+ \qquad \text{for } i = 1, \ldots, f \tag{6.6}$$

In ILP1 δ, k_i are decision variables, $\lambda_j, L_j, u_{ij}, Q$ are constants. The greatest divergence δ from the required ship length frequency is minimized. Constraint (6.3) guarantees that the chosen berth lengths fit on the quay and no space is left unused. Equation (6.3) can be satisfied with equality because λ_1 divides Q. By constraint (6.4) there is at least one berth at least as long as the longest ship. Value $\sum_{\{i:\lambda_i \geq L_j\}} u_{ij} k_i L_j / Q$ in (6.5) is the fraction of quay length at which ships of length L_j can be positioned. On the left-hand-side of (6.5) the summation

over $j = \ell, \ldots, e$ and $\lambda_i > L_j$ represents the fact that vessels of length $L_\ell < \cdots < L_e$ may be positioned at the same berths. For the same reason δ is multiplied by $(e - \ell + 1)$ on the right-hand-side of (6.5). Thus, constraints (6.5) set the distance δ between the required importance of vessel length $\ell, \ldots, e$ and fraction of the space at the quay obtained by these lengths.

In an alternative integer linear program referred to as **ILP2** the biggest difference between the required value q_j of availability and the actual fraction of the quay length where ships of length L_j can be moored, is bounded for each length separately. Then, the following constraints:

$$q_j - \sum_{\{i:\lambda_i \geq L_j\}} u_{ij} k_i L_j / Q \leq \delta \qquad \text{for } j = 1, \ldots, e \tag{6.7}$$

are added to (6.2)-(6.5). Note that various combinations of vessel sizes can simultaneously be positioned at the quay and the difference between the sum of vessel size importances in the combination and the fraction of the quay obtained by such a combination should be minimized. Since the number of possible vessel size combinations grows exponentially with e including constraints like (6.5), (6.7) for any vessel combination that can possibly moore at the quay is impractical. Hence, we remained with only two ILP versions.

Now let us discuss obtaining the importance histograms. In Chapter 5 ship size classes i were introduced with their relative frequencies a_i, p_i/L_i distributions and upper ends of ship lengths in a class. The length range upper ends can serve as vessel lengths L_i in the above formulations. A method matching class frequency a_i histogram by use of ILP1 or ILP2 will be referred to as ILP1f and ILP2f, respectively. A histogram of area in time×space for class i occupation can be built by assuming $q_i = \overline{p_i/L_i} * L_i^2 / \sum_{j=1}^{7} (\overline{p_j/L_j} * L_j^2)$, where $\overline{p_i/L_i}$ is estimation of service time per unit of vessel length for class i. A method matching this kind of histogram will be called area matching and referred to as ILP1a and ILP2a for ILP1 and ILP2, respectively.

6.3.3 Big Berths First

Big Berths First algorithm (BBF in short) builds as many as big berths as possible. The algorithm proceeds from the longest berths λ_f to the shortest λ_1. For the considered berth length λ_i and the remaining quay size Q', set $k_i = \lfloor Q'/\lambda_i \rfloor$ and $Q' = Q' - k_i \lambda_i$, for $i = f, \ldots, 1$.

6.3.4 Hill Climber

Local search methods improve solutions by gradually modifying them. Such trajectory-based metaheuristics are particularly suited when solution evaluation costs are high, as it is in our case. Local search methods need explicit representation of the solution, the set of admissible solution changes called moves, a starting solution and a strategy to choose moves. The solutions are represented here as berth length frequency vectors K. There are two types of moves: split and merge. In the split move, berth of length λ_i is split into two shorter berths of length λ_j, λ_k such that $\lambda_j + \lambda_k = \lambda_i$. All such pairs j, k are tested. In the merge move, for a pair of lengths λ_j, λ_k the pair is substituted by the berth length λ_i satisfying $\lambda_j + \lambda_k = \lambda_i$. Again, all pairs j, k are tested. Algorithm BBF provides the starting solution. The set of all solutions attainable by executing moves on the current solution are called neighborhood. Hill climber (HC for short) is a local search method which proceeds to the first-found better solution in the neighborhood and stops if no improving neighbor exists.

6.3.5 Tabu Search

Tabu search is also a local search method which extends a hill climber by use of memory to prevent accessing the already visited solutions. To this goal information is stored in a queue called tabu list. In our implementation indices of executed moves are stored in the tabu list. A move stored in the tabu list can be undone only if it improves the best solution found. Except for the tabu list, HC features are inherited. For practical usability, our tabu search algorithm is limited to 401 evaluations of the objective (6.1) and at most 24h runtime, including evaluations.

6.3.6 Random

This algorithm adds berths of random length to solution K until exhausting the remaining free quay length. This procedure is repeated 10 times and will be referred to as *Rnd10*. An average $F(K)$ from all generated layouts is returned. No particular solution of RND10 will be used. Rnd10 is a reference algorithm for other algorithms performance comparisons.

Table 6.4: Overview of BF results

Le Havre STM, real traffic in 2016: 2271 arrivals					
intensity ι [a/w]	20	50	100	200	500
$\min_s\{n_s\}$	1052	2610	5214	10409	26016
$\max_s\{n_s\}$	1085	2657	5259	10452	26051
$F(K^*)$	25.24	25.89	409.21	4664.34	17438.73
$F(K^*)+\sigma(K^*)$	25.91	26.32	607.73	5079.50	18373.24
No.of sol. in $[F(K^*), F(K^*)+\sigma(K^*)]$	1180	79	27	32	23
No.of BF solutions	2052	2052	2052	2052	2052
Shanghai STM, real traffic in 2016: 11606 arrivals					
intensity ι [a/w]	20	50	100	200	500
$\min_s\{n_s\}$	1093	2639	5229	10430	26034
$\max_s\{n_s\}$	1148	2709	5286	10518	26127
$F(K^*)$	25.26	24.17	25.32	27.70	4197.84
$F(K^*)+\sigma(K^*)$	29.70	27.86	27.90	31.37	4579.60
No.of sol. in $[F(K^*), F(K^*)+\sigma(K^*)]$	8880	8200	6455	749	109
No.of BF solutions	8958	8958	8958	8958	8958

a/w – arrivals per week
K^* – partition with the minimum score (6.1), i.e., the best one

6.4 Features of SQPP Solutions

In this section we analyze features of the SQPP solutions obtained by brute force partition enumeration. Thus, optimum partitions are known. The test setting follows the description in Chapter 5 and 6.2. In more detail, two STMs representing Le Havre and Shanghai were used. For Le Havre quay length was $Q = 3500$m and for Shanghai it was set to $Q = 5000$m. In both cases the set of admissible berth lengths was $B = \{100, 200, 300, 400, 500, 600\}$ meters. The numbers of nondominated partitions were 2052 for Le Havre and 8958 for Shanghai, respectively, as shown in Tab.6.3. One year traffic was generated with intensity levels 20, 50, 100, 200, 500 a/w. For each intensity level, 10 test scenarios were generated. Thus, each partition was evaluated on 5 arrival intensity levels, each with 10 test scenarios s, solved by 8 BAP scheduling algorithms i, as described in Section 6.2. This created, for each partition and intensity level, a population of 80 evaluations. $F(K)$ as defined in (6.1) and standard deviation of $F(K, s, i)$ were calculated for each such population. The results are shortly outlined in Tab.6.4. In this table the range of vessel numbers in the generated test scenarios ($\min_s\{n_s\}$, $\max_s\{n_s\}$), range of $MWFT$s from the best partition score $F(K^*)$ to the best partition plus one standard deviation $F(K^*)+\sigma(K^*)$, number of partitions in range $[F(K^*), F(K^*)+\sigma(K^*)]$ and the total number of verified partitions are given.

It can be concluded from the the results in Tab.6.4 that traffic intensities of 20 arrivals per week (a/w) for Le Havre, and up to 100 a/w for Shanghai, are easy to handle. This is indicated by low and similar values of the best $F(K^*)$ in these arrival intensity ranges. Furthermore, the number of partitions in range $[F(K^*), F(K^*) + \sigma(K^*)]$ near the smallest $F(K^*)$, is large (especially in Shanghai). It means that at this low traffic it is easy to schedule the arriving vessels on nearly any quay partition (except for the degenerate solutions) because there is an excess of available berthing space, quay partitioning is actually immaterial. Hence, there are many partitions which are nearly as good as the best one. We will discuss structure of the good and the particularly bad partitions in sections 6.4.1 and 6.4.2. Intensities 50 a/w in Le Havre and 200 a/w in Shanghai are border cases because low values of $F(K^*)$ are achievable, but only by a small set of partitions. Let us note that the 2271 arrivals that actually happened in Le Havre in 2016 is roughly intensity 42 a/w, and the 11606 arrivals in Shanghai is roughly 223 a/w. Thus, both ports seem to be close to the border traffic intensities, and schedules for the arriving vessels are not hard to construct on reasonably chosen quay partitions. With even greater arrival intensity, the number of partitions with their $F(K)$ score close to the best one tends to decrease. Thus, quay partitioning matters in vessel quality of service.

6.4.1 Patterns in Berth Length Selection

In Fig.6.2 general tendencies in the selection of berth lengths are shown. In the pictures berth lengths from set $B = \{100, 200, 300, 400, 500, 600\}$ are shown along the horizontal axis. The average number of certain length berths in some subset of solutions is shown along the vertical axis. The standard deviation of the chosen numbers of certain berth length is also shown. Three types of partitions are shown in Fig.6.2. The leftmost column displays the 20 best partitions, the central column the partitions in the range $[F(K^*), F(K^*) + \sigma(K^*)]$, where $F(K^*)$ is the score of the best partition and $\sigma(K^*)$ a standard deviation of its evaluations. The rightmost column displays berth length frequencies in the 50 worst solutions. STMs and vessel traffic intensities are shown in rows. The first two rows show results for the Le Havre STM, and the last two rows comprise results for the Shanghai STM. Light traffic (20 a/w) is depicted in the first and the third picture rows. Intensive traffic (500 a/w) is presented in the second and the fourth rows. It can be observed that in general the best solutions (left column) tend to have many large berths (i.e. capable of hosting the longest vessels with $L_{max} = 400$m). Note, however, that in these

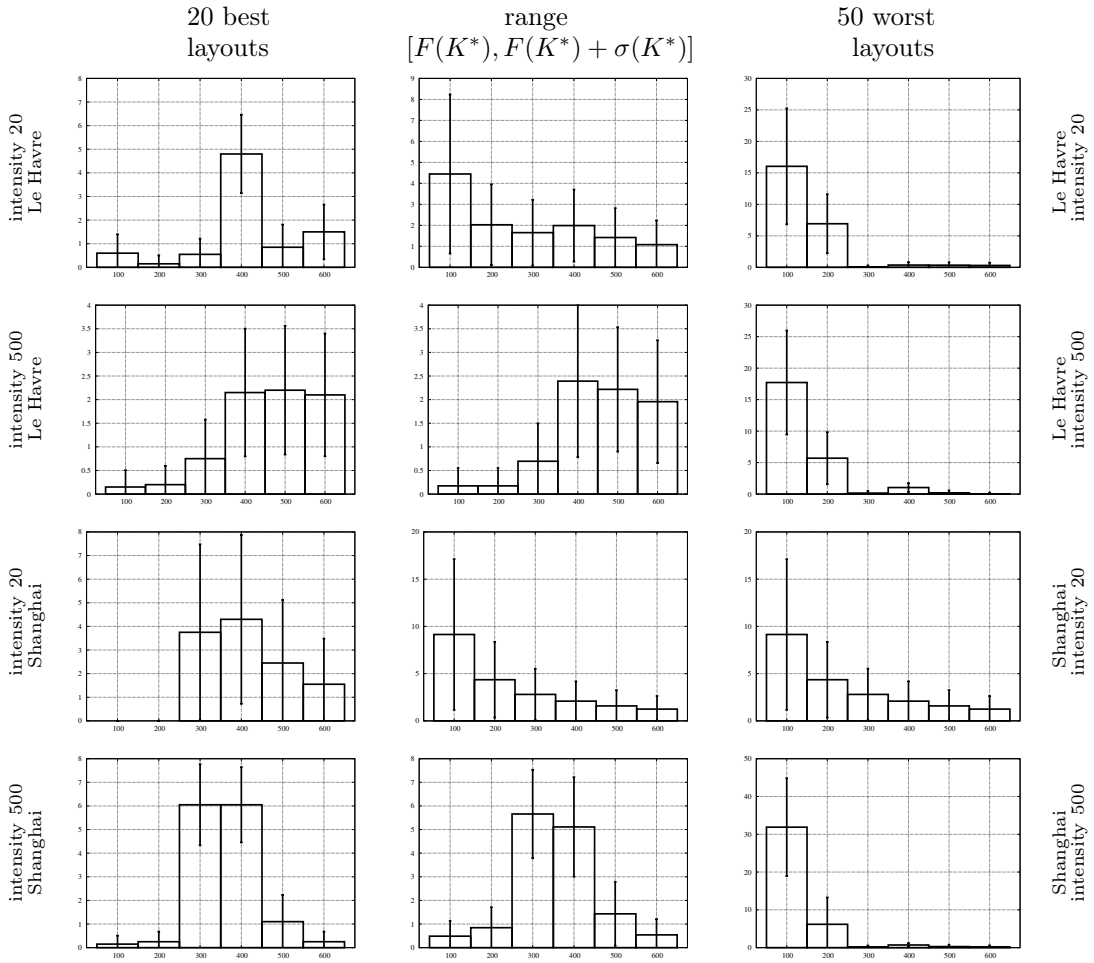


Figure 6.2: Numbers of berth lengths from set $B = \{100, 200, 300, 400, 500, 600\}$ meters in subsets of partitions for Le Havre, Shanghai, intensities 20 a/w and 500 a/w.

solutions (left column), not one but many berth lengths are chosen yet in various frequencies. Conversely, the worst partitions have many short (100m) and very few large berths, which seems the perfect worst solution. These general observations are moderated by traffic intensities and STMs. In very low traffic (Fig.6.2 first and third row) there are many partitions in the range $[F(K^*), F(K^*) + \sigma(K^*)]$ (cf. Tab.6.4), performing relatively well and the strength of the distinction between good and bad solutions in this situation is small. Consequently the numbers of berths chosen in the population defined by range $[F(K^*), F(K^*) + \sigma(K^*)]$ for low traffic, represent frequencies of berth lengths in the whole set of partitions rather than the

frequencies of berth lengths in good solutions. If we select even more restrictively the 20 best solutions at light traffic (left column, first and third row) then tendencies in good solutions become more visible: In Le Havre berths with $L_{max} = 400$m are most frequently chosen, in Shanghai berth lengths 300m and longer are selected. The difference between Le Havre and Shanghai can be explained by different mix of vessel lengths in the STMs: Shanghai has more short vessels than Le Havre. For the intensive traffic (Fig.6.2 second and fourth row) for the population of solutions in range $[F(K^*), F(K^*) + \sigma(K^*)]$ (central column) the results resemble the restricted set of the 20 best solutions (left column). It can be concluded that for intensive traffic, good partitions have to be selected carefully, their number is small (see Tab.6.4) and hence they are more like the 20 best solutions. The chosen numbers of certain length berths differ in Le Havre and Shanghai because their STMs differ in the mix of long and short vessels. In Le Havre large berths ($\lambda_i \geq 400$m) dominate in the best solutions because Le Havre is the first oceanic port in western Europe and large mother ships are quite frequent. In Shanghai, berth lengths 300m and 400m are the most frequently selected because Shanghai Yangshan port is located in the estuaries of two large rivers and short ships are more common than in Le Havre [90]. Note that in both ports not one but a few berth lengths are chosen in the 20 best solutions.

6.4.2 Quality Dispersion

In Fig.6.3 dispersion $\sigma(K)$ vs solution quality $F(K)$ for all partitions is shown. Along horizontal axes $F(K)$ is shown. Standard deviation $\sigma(K)$ of $F(K, s, i)$ in the population of 80 evaluations is shown along the vertical axes. Each point represents one partition. Colors show the number of large berths, that is $|\{\lambda_i : \lambda_i \geq 400\text{m}\}|$. The choice of $F(K)$ as a partition quality measure (horizontal axis) is natural because $F(K)$ is a quality of service measure. Let us note that partitions incurring large dispersion of service quality are hard to manage: there are scenarios where such partitions perform well but also scenarios where quality of vessel service is low, there are scheduling algorithms which manage the scenarios on such partitions well, but also algorithms that are not equally effective. Thus, standard deviation of $F(K, s, i)$ is also a useful indicator of the partition goodness, and partitions with low dispersion should be preferred. In Fig.6.3a,b results for low intensity traffic, and in Fig.6.3c,d for high intensity traffic, are shown, respectively. It can be observed that differences between good and bad partitions can be large, even two orders of magnitude differences both in $F(K)$ and in the standard deviation

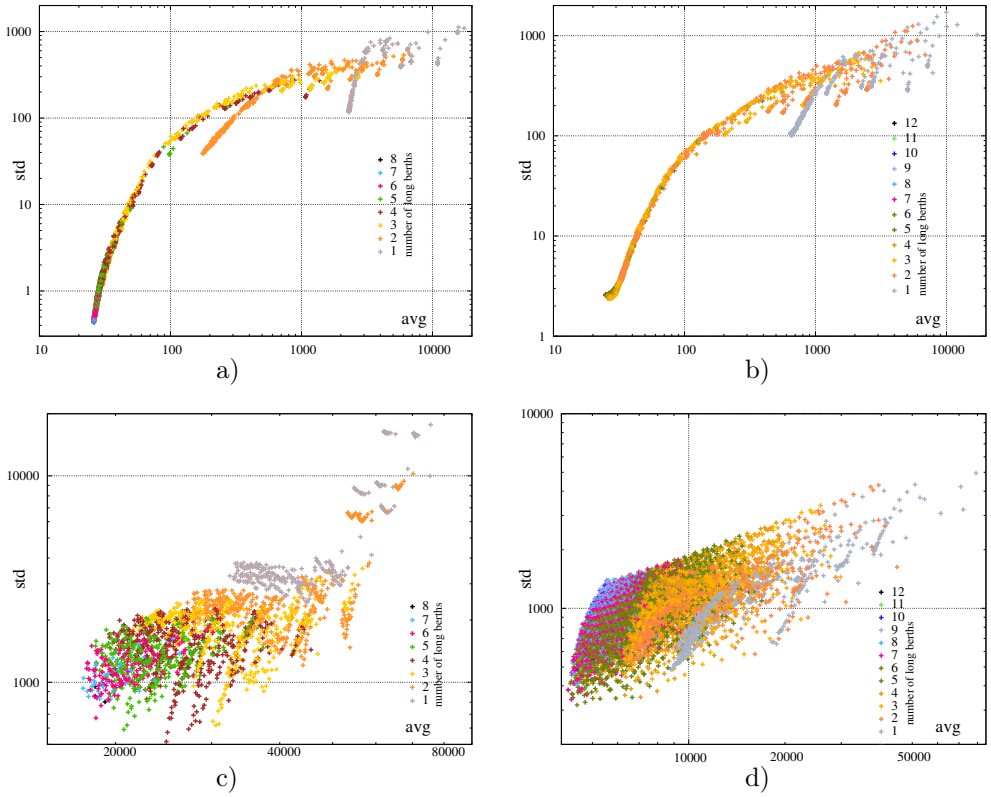


Figure 6.3: $MWFT$ average–standard deviation diagram including all brute-force solutions. a) Le Havre intensity 50, b) Shanghai intensity 100, c) Le Havre intensity 500, d) Shanghai intensity 500 a/w.

are possible. Thus, optimization of quay partition is relevant. Generally, partitions with high quality of service (low $F(K)$) are also the partitions with low $\sigma(K)$. This is particularly true for low intensity traffic (Fig.6.3a,b) – by improving $F(K)$ also $\sigma(K)$ is improved. In the case of low intensity traffic many points overlap in Fig.6.3a,b, which indicates that there are many solutions similar in the criteria of $F(K)$ and $\sigma(K)$. This observation is in line with the earlier find (cf. Tab.6.4) that at low traffic there are many similar solutions, in particular similar to the one with best $F(K)$. Colors in Fig.6.3a,b show that partitions with small number of large berths are particularly bad because they do not differ much from the worst-case solution comprising one berth of length $L_{max} = 400$m and all remaining berths as short as possible (100m).

For high intensity traffic (Fig.6.3c,d) similar observations can be made: while improving

Table 6.5: $MWFT$ average–standard deviation nondominated solutions at ι =500 a/w

Le Havre			
K	$F(K)$	$\sigma(K)$	No.large berths
(0,0,0,2,3,2)	17438.7	934.5	7
(0,0,0,1,5,1)	17762.7	904.2	7
(0,0,1,4,2,1)	17847.7	852.7	7
(0,0,2,1,5,0)	18411.3	671.2	6
(0,2,2,0,5,0)	20713.2	590.4	5
(1,4,2,0,4,0)	24781.3	515.8	4
Shanghai			
K	$F(K)$	$\sigma(K)$	No.large berths
(0,0,6,8,0,0)	4197.8	381.8	8
(0,0,7,6,1,0)	4202.1	380.3	7
(1,0,7,7,0,0)	4297.4	340.2	7
(2,0,8,6,0,0)	4477.1	318.8	6

$B = \{100, 200, 300, 400, 500, 600\}$ admissible berth lengths in meters

$F(K)$ also dispersion of $F(K, S, i)$ is decreasing, solutions with inadequate number of large berths are the worst. However, in high intensity traffic the partitions are less similar to each other, in the performance sense, which is marked by more dispersed cloud of points showing the solutions. If $F(K)$ and $\sigma(K)$ are two criteria of partition quality, then it can be observed that a Pareto-front of nondominated solutions exist when traffic is intensive. The nondominated solutions in Fig.6.3c,d are presented in Tab.6.5. Beyond the values of $F(K), \sigma(K)$, also the partitions and the number of large berths are shown in Tab.6.5. For example, partition $(0, 0, 0, 2, 3, 2)$ for Le Havre has 2 berths of length 400m, 3 berths of length 500m, and 2 berths 600m-long. It can be observed that in Le Havre partitions with the smallest $F(K)$, all berths can accommodate the longest vessels (L_{max} = 400m). However, this impacts dispersion of $F(K, S, i)$ because traffic of large and short vessels interfere on these large berths. Adding a few shorter berths allows to slightly separate short vessels from the large ones. Thanks to this, standard deviation of $MWFT$ decreases, yet $F(K)$ increases. In Shanghai similar phenomena emerge, yet the traffic of shorter vessels in Shanghai is more intensive (in relation to large vessels) than in Le Havre, and consequently the partition with the smallest $F(K)$ already has berths that are shorter than L_{max}.

Table 6.6: The best solution dominance in Student's t-test

Le Havre					
intensity ι [a/w]	20	50	100	200	500
$p_{1\to 2}$	0.037	0.343	9.31E-23	1.46E-12	0.159
No.sol.$p_{1\to i} \geq 0.01$	6	2	0	0	1
Shanghai					
intensity ι [a/w]	20	50	100	200	500
$p_{1\to 2}$	nan	nan	0.655	0.394	0.462
No.sol.$p_{1\to i} \geq 0.01$	nan	nan	23	3	1

$p_{1\to i}$ – p-value for the test that the 1st- and ith-best solutions perform the same
No.sol.$p_{1\to i} \geq 0.01$ – number of solutions with $p_{1\to i} \geq 0.01$
nan – over 800 best solutions are indistinguishable in quality, p-values couldn't be computed

6.4.3 Solution Similarity

In this section we analyze probabilities that some solutions of SQPP turned out good by random coincidence. This analysis can be also rendered in terms of the SQPP solutions similarity. More formally, for pairs of partitions K_1, K_2 probabilities (p-values) in Student's t-test for paired samples are calculated with the null hypothesis that the performance of the two partitions is the same. The samples are paired in this sense that the results in 80 evaluations $F(K_1, s, i), F(K_2, s, i)$ over the set of 10 scenarios s and 8 algorithms i are matched. If the obtained p-values are low then it means that the two partitions performed differently in the evaluations and the dominance of one solution over the other is not a coincidence. Conversely, high p-values indicate that two partitions K_1, K_2 performed similarly in the evaluations, the order of the two solutions with respect to $F(K_1), F(K_2)$ could easily be different. The p-value is not only a score allowing to reject a null hypothesis, but can also be interpreted as a measure of similarity between the two solutions K_1, K_2. For brevity of exposition we will refer to the p-values as an indicator of similarity.

In Tab.6.6 dominance of the best partition is demonstrated using its p-values in the test with the second-best solution (denoted $p_{1\to 2}$). It can be seen that the first and the second-best solutions are often similar in performance because the p-values happen to be greater than 0.3. The Le Havre case at $\iota \in \{20, 100, 200\}$ is different because probabilities $p_{1\to 2}$ are at most 0.037. In the case of Shanghai and $\iota \in \{20, 50\}$ the number of solutions close to the best one is large (see Tab.6.4) so that over 800 best solutions were indistinguishable in their evaluations. Consequently, p-values couldn't be calculated. Despite this computational inconvenience, this

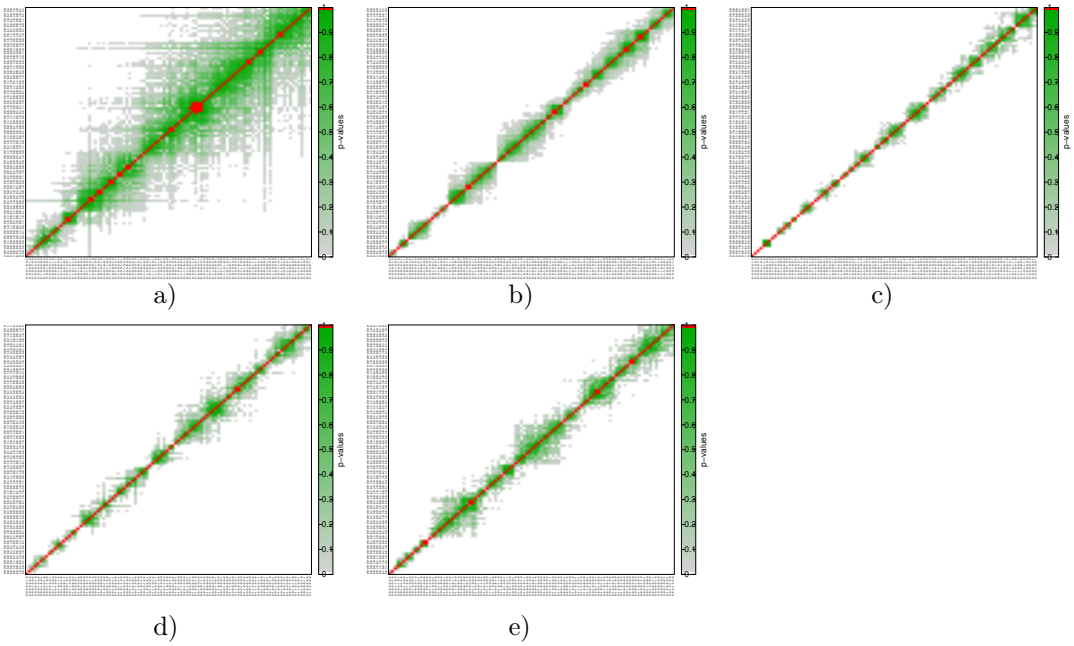
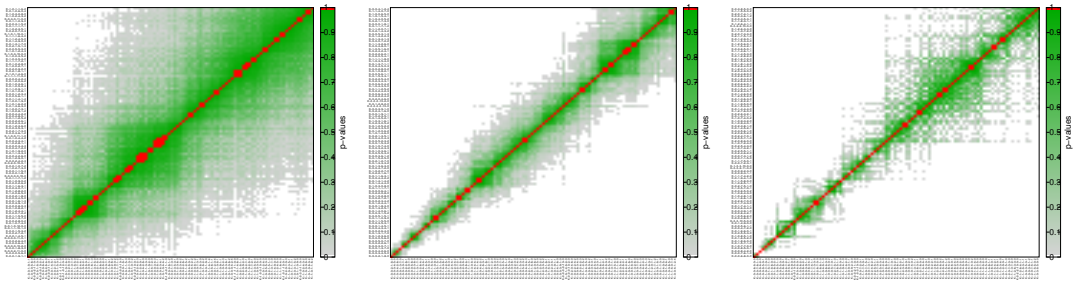


Figure 6.4: p-values of a t-test that pairs of partitions perform the same. 100 best partitions, Le Havre, arrival intensity a) 20, b) 50, c) 100, d) 200, e) 500 a/w.

situation shows that at low intensity traffic best solutions for Shanghai are also very similar in the performance sense.

The number of solutions which p-values are at least 0.01 in the paired test with the best solution is also shown in Tab.6.6. It can be concluded that the set of solutions similar to the best one is not large and its cardinality decreases with arrival intensity.

The results of pairwise similarity tests for the 100 best solutions are shown in Fig.6.4 for Le Havre and in Fig.6.5 for Shanghai. The partitions K are put along the axes in the order of increasing $F(K)$. The p-values for pairs of partitions are indicated in color: values smaller than 0.01 are shown as white points, values greater than 0.99 are marked red, values in range [0.01,0.99] are shown as shades of green. It can be seen in Fig.6.4 and Fig.6.5 that the tendency of forming groups of solutions performing similarity depends on the traffic intensity. In Le Havre the solutions differ most for arrival intensity 100 a/w (Fig.6.4c). Blocks of similar solutions can be seen both for the lighter (Fig.6.4a,b) and for more intensive traffic (Fig.6.4d,e). For the light traffic there is abundant free space at the quay and it is possible to build good schedules on



a) b) c)

Figure 6.5: p-values of a t-test that pairs of partitions perform the same. 100 best partitions, Shanghai, arrival intensity a) 100, b) 200, c) 500 a/w.

many alternative partitions. Hence, groups of similarly performing partitions almost naturally exist. This situation echoes results shown in Tab.6.4. In heavy traffic space at the quay is over-subscribed and vessels inevitably have to wait. Some groups of partitions perform similarly because they do not offer any alternatives to the waiting. The border case of arrival intensity 100 a/w (Fig.6.4c) distinguishes solutions the most because traffic is intensive enough to fill the available quay and it is also light enough to allow avoiding vessel wait by the virtue of a good partition choice. Similar observations can be made for Shanghai (Fig.6.5). For arrival intensities 20, 50 a/w over 800 best solutions were indistinguishable from the performance point of view and p-values couldn't be computed (though should be considered equal to 1). For arrival intensities greater than 100 and growing, the solutions become increasingly dissimilar.

It can be concluded that the set of solutions with similar performance pattern as the best solution is not large (cf. Tab.6.6), and its size decreases with traffic intensity. Groups of similarly performing solution appear mostly at light traffic (Fig.6.4a, Fig.6.5a) and to a lesser extent also at heavy traffic (Fig.6.4e, Fig.6.5c).

6.4.4 Equipartition

By *equipartition* we understand a type of solution in which the quay is divided into equal length berths. The chosen berth length λ_i must be at least as long as $L_{max} = 400$. Since the chosen berth lengths λ_i may not divide Q, the remaining quay length is assigned to as long berth lengths as possible. Note that the BBF algorithm proposed in Section 6.3.3 builds only one of possible

Table 6.7: Equipartition solutions for intensity 500 a/w

Le Havre			
K	$F(K)$	$\sigma(K)$	No.large berths
(0,0,0,0,1,5)	18086.3	1499.4	6
(0,0,0,0,7,0)	18792.4	882.8	7
(0,0,1,8,0,0)	19726.5	1085.7	8
Shanghai			
K	$F(K)$	$\sigma(K)$	No.large berths
(0,1,0,0,0,8)	6494.7	1465.6	8
(0,0,0,0,10,0)	5549.0	1385.4	10
(0,1,0,12,0,0)	4695.9	671.1	12

$B = \{100, 200, 300, 400, 500, 600\}$ admissible berth lengths in meters

equipartition solutions. The performance of equipartition layouts for arrival intensity 500 a/w is shown in Tab. 6.7. With respect to $F(K)$ equipartition solutions are not optimum, but they are not among the worst either (cf. Tab.6.5 vs Tab.6.7 and Fig.6.3). However, their solution quality dispersion score $\sigma(K)$ is significantly worse than for the solutions on $F(K), \sigma(K)$ Pareto-front. In conclusion, equipartition is not recommended as better solutions are readily available.

6.5 Evaluation of the Partitioning Algorithms

In this section we report on the performance of the algorithms for SQPP. The algorithms were evaluated on the STM, Q pairs: Hamburg STM, $Q = 7500m$, Le Havre STM, $Q =$ 3500m, Rotterdam STM, $Q = 6000m$, Shanghai STM, $Q = 5000$m, Singapore STM, $Q =$ 14000m, Singapore STM, $Q = 20000$m, Singapore STM, $Q = 26000$m, with admissible berth lengths $B = \{100, 200, 300, 400, 500, 600\}$ meters and for arrival intensities $\iota \in \{20, 50, 100,$ $200, 500, 1000\}$ a/w. The methods of scenario generation were the same as described in Section 5.4. Due to high complexity, only the instances of Le Havre $Q = 3500$m, Shanghai $Q = 5000$m for intensities $\iota \in \{20, 50, 100, 200, 500\}$ were solved by brute force approach.

Algorithms can be evaluated on the criteria of computational cost and solution quality. The computational cost has two components: algorithm own runtime and the cost of solution evaluations. Generally the former is negligible compared with the latter. For example, histogram matching methods from Section 6.3.2 were executed in at most 0.02s on a standard PC with i7-8550U@1.80GHz CPU, while the partition evaluations on a high-performance server CPU can last over 751s (see Tab.6.2). Furthermore, computational effort of the evaluations can

Table 6.8: Example number of evaluations by hill climber (min,median,max)

STM, Q \ ι	20	50	100	200	500	1000
Le Havre, Q =3.5km	6,6,9	6,6,6	6,6,6	6,6,6	14,14,14	8,8,8
Shanghai, Q =5km	5,6,12	5,6,10	10,10,51	69,69,69	60,60,60	60,60,60
Singapore, Q =14km	5,5,5	5,5,15	5,6,17	30,35,38	182,190,203	60,60,65
Singapore, Q =26km	5,5,5	6,6,13	5,13,16	5,5,15	89,99,131	51,51,55

Table 6.9: Example number of evaluations by tabu search (min,median,max)

STM, Q \ ι	200	500	1000
Le Havre, Q =3.5km	401,401,401	317,392,401	66,75,129
Shanghai, Q =5km	270,299,401	146,146,315	98,101,113
Singapore, Q =14km	113,401,401	319,319,401	54,60,60
Singapore, Q =26km	401,401,401	401,401,401	137,154,154

be arbitrarily resized by changing the number of test scenarios and the size of the scheduling algorithms portfolio. Hence, time performance of the algorithms should rather be assessed by the number of calls to the solution evaluation procedure. For this complexity measure algorithms from Section 6.3 can be ordered as follows: histogram matching and BBF have cost of one evaluation, Rnd10 has cost of 10 evaluations. Example numbers of evaluations of hill climber are shown in Tab.6.8 and of tabu search in Tab.6.9. Thus, the hill climber used from 5 to 203 evaluations, and tabu search used 54 to 401 evaluations (the fixed maximum value), varying with the STM, Q, and arrival intensity. Brute force enumeration has the highest evaluations cost as shown in Tab.6.3. This cost is growing polynomially with the Q/λ_i ratios but exponentially with the number f of admissible berth lengths. Returning to Tab.6.8 and Tab.6.9, HC and tabu search were run 5 times to collect the data for these tables. It could be observed that for small intensities $\iota \leq 200$a/w tabu stops on the iteration limit, while for larger intensities on 24 hour runtime limit.

The second algorithm performance measure is quality of the solutions as defined in (6.1). As observed in Section 6.4, for low arrival intensities quay partitioning is immaterial and any partition is equally good. Consequently, differences between solutions built by the algorithms are negligible if arrival intensity is low. For this reason we do not report on the cases where the differences in solution scores are insignificant. Precisely, the results are not reported, as not distinguishing the algorithms in a significant way, if $(\max_{a\in\mathcal{A}}\{F(K_a)\} - \min_{a\in\mathcal{A}}\{F(K_a)\}) < \min_{a\in\mathcal{A}}\{\sigma(K_a)\})$, where $\mathcal{A}$ is the set of algorithms and K_a is a solution constructed by $a \in \mathcal{A}$. Quality of the results including brute force approach are presented in Tab.6.10, and for larger

Table 6.10: $F(K)$ for algorithmically built SQPP solutions

Le Havre, $Q = 3500$m									
ι	BF	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
50 a/w	25.945	**25.945**	25.959	25.959	25.959	26.082	28.283	**25.945**	**25.945**
100 a/w	409.17	**409.17**	613.12	548.60	612.87	675.55	1403.85	**409.17**	**409.17**
200 a/w	4664.4	4778.5	4924.2	4804.5	4925.0	5101.4	6464.9	4778.5	**4664.4**
500 a/w	17437	18085	18353	17851	18351	18343	22619	17770	**17437**
Shanghai, $Q = 5000$m									
ι	BF	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
200 a/w	27.689	50.952	29.337	29.320	31.743	31.741	36.371	27.976	**27.689**
500 a/w	4199.1	6492.6	4693.5	4201.8	4879.9	4880.7	5377.8	4433.1	**4199.1**

bold – the best solution by a *non-brute force* algorithm

intensities in Tab.6.11 (excluding BF). For brute force method the optimum value $F(K^*)$ is shown. For Rnd10, average of the 10 random solutions are shown under $F(K)$. For the hill climber and tabu search the median of $F(K)$ in the run repetitions is shown. The best non-brute force algorithm results are shown in boldface.

It can be seen in Tabs 6.10 and 6.11 that for low arrival intensities (Le Havre, $\iota = 50$, Singapore Q=26000m and Q=20000m , $\iota = 500$) quality of the solutions does not differ much. For larger intensities, our algorithms have solution values significantly below Rnd10. If the optimum solution K^* is known (Tab.6.10), then there are algorithms with solutions values close to $F(K^*)$. This signifies that our methods indeed work and provide good solutions. For the cases where tabu search didn't build the best solutions, ILP2 is the dominating algorithm, with IPL2f being best once and ILP2a in 4 other cases. BBF has more complex behavior because for Le Havre it builds the best heuristic solutions at intensities $\iota \leq 100$ a/w. For Le Havre at $\iota = 500$ a/w BBF is only 3.7% worse than tabu search which provided the best solution in this case. For other STMs BBF fails and is even worse than Rnd10. Such variable BBF performance occurs because BBF builds partitions with as long berths as possible, which is effective in Le Havre with dominating long mother ships, but is counterproductive when shorter vessels are very common.

Again, it may be argued that the above results were obtained by a random coincidence. In order to verify this, p-values in the paired samples Student's t-test were calculated for each pair of the algorithms with the null hypothesis that the performance of the two algorithms is the same. The samples are paired in this sense that results $\{F(K_a), F(K_b)\}$ of a pair $\{a, b\}$ of algorithms on the same instance (STM, ι, Q, B) can be matched and compared. Conventionally, the results

Table 6.11: $F(K)$ for algorithmically built SQPP solutions

Hamburg, Q =7500m								
ι	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
1000 a/w	18347	10988	9314.9	10988	10991	13316	9282.7	**9202.3**
Le Havre, Q =3500m								
ι	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
1000 a/w	38774	38776	37894	38785	38791	44093	38436	**37249**
Rotterdam, Q =6000m								
ι	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
1000 a/w	24981	21823	**21300**	21903	21782	23964	22677	21941
Singapore, Q =14000m								
ι	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
500 a/w	418.84	116.36	69.76	116.32	54.66	224.73	49.88	**46.81**
1000 a/w	5082	3903.3	3884.6	3903.6	**3881.5**	4530.4	4174.3	4174.3
Singapore, Q =20000m								
ι	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
500 a/w	21.112	20.296	20.113	20.298	20.049	20.477	20.033	**19.990**
1000 a/w	2331.4	1556.5	1438.3	1556.6	**1400.4**	1846.1	1730.0	1730.0
Singapore, Q =26000m								
ι	BBF	ILP1f	ILP2f	ILP1a	ILP2a	Rnd10	HC	tabu
500 a/w	19.638	19.569	19.555	19.569	**19.547**	19.579	19.559	19.558
1000 a/w	820.91	310.07	215.06	310.25	**180.22**	371.05	312.10	222.29

bold – the best solution found

of the Student's t-test are used used as an indicator whether a null hypothesis can, or cannot, be rejected at certain confidence level. Yet, for brevity of presentation we will present the p-values in terms of algorithm similarity measure. Low p-values indicate that probability of obtaining the given dominance between by two algorithms results as a random coincidence is low. Hence, low p-values signify that the two considered algorithms perform differently. The results of the paired algorithm test for similarity are presented in Fig.6.6a. For many algorithm pairs p-values are in a range of (0.1,0.2) which is larger than confidence levels of 0.05 or 0.1 conventionally assumed to reject a statistical hypothesis. Yet, it is hard to assess whether probabilities in range (0.1,0.2) are high or low. Consider for example the p-value for brute force (BF) and Rnd10 pair. This pair of algorithms may be thought of as different because BF provides the optimum solution while Rnd10 solutions may be considered unbiased, i.e. tending neither toward good nor toward bad solutions. But even this algorithm pair has p-value=0.1403. Therefore, it is plausible to assume that algorithm pairs with p-values smaller than 0.1403 may be considered at least as different as BF-Rnd10 pair. The results scaled to p-value=0.1403 as a unit, are shown in Fig.6.6b. It can be seen that in most of the cases the algorithms perform at least as differently as {BF,Rnd10} pair. {ILP2a,ILP2f} are the most similarly performing methods. This is intuitively expected because in 75% of tested instances the difference between solutions of ILP2a and ILP2f was less than

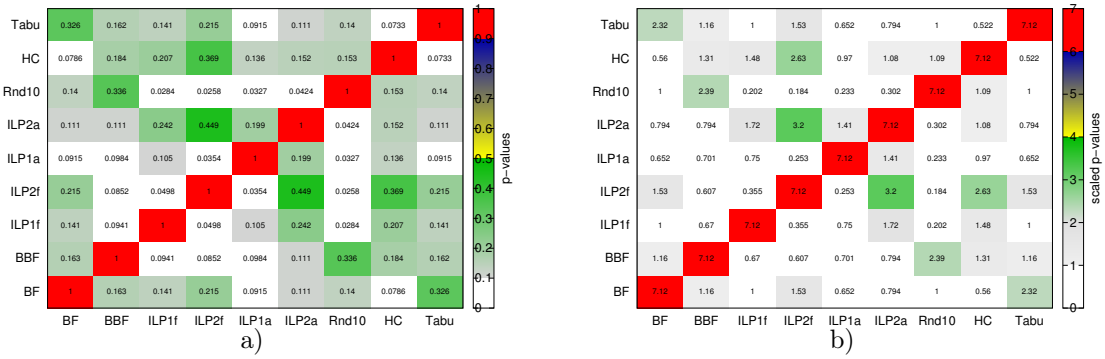


Figure 6.6: Algorithm similarity in Student's t-test. a) p-values, b) p-values scaled to BF vs Rnd10 p-value. Higher values indicate more similar algorithms.

2.7%. The next most similarly performing pair is {ILP2f,HC}. Also in this case, in majority of instances results are quite close to each other: in 75% of instances the solution quality difference is below 4.3%. Thus, generally our algorithms are distinct methods performing differently, yet there are instances on which some pairs obtain similar results.

This section can be concluded by observing that the presented algorithms have their own domains of good performance: BBF is good for ports with many large vessels (e.g. Le Havre). ILP2f and ILP2a can be recommended when traffic is intensive and solution evaluations are costly. Tabu search is good in all cases, but its main limitation is high solution evaluation cost.

6.6 Technical aspects of solving SQPP

In this section, SQPP solving method will be presented. Time scalability and reliability of the SQPP optimization workflow on the SLURM platform will be analyzed. The tests were executed on SLURM version 22.05.0 of Eagle/Altair supercomputer at Poznań Supercomputing and Networking Center [85] on the hardware platform described in Section 6.2. All the scalability tests on configurations of the workflow mentioned in this section were executed at least three times. Mind, that some parts of the workflow were executed many more times because the partition evaluation is a repetitive process. The results presented in this section were obtained for Le Havre STM. The description will be based on the example of Hill Climber (Section 6.3.4). For other algorithms the core of the workflow is the same.

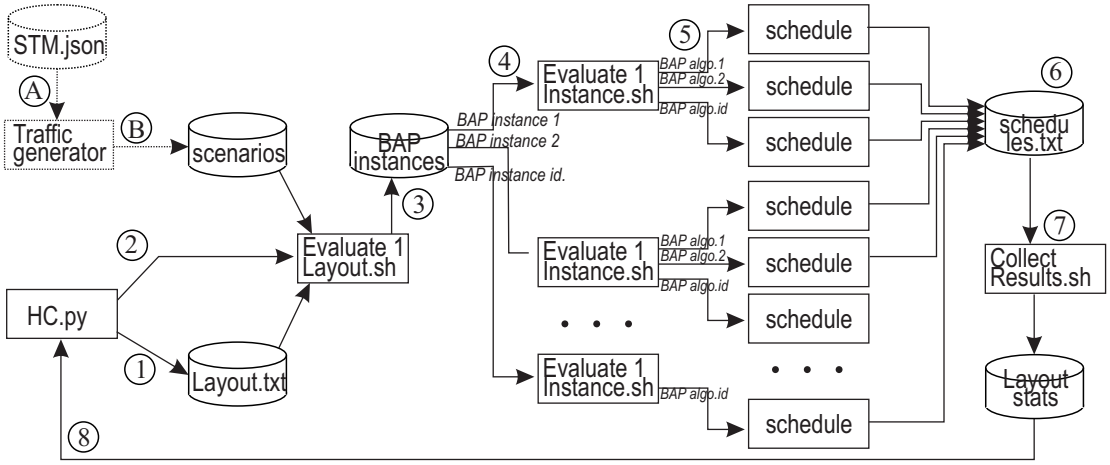


Figure 6.7: Workflow architecture.

6.6.1 Optimization Workflow

Since we had an existing code base for BAP, and hence some code heritage, we decided to implement the optimizer as a workflow wrapping the existing code instead of refactoring it into a monolithic solution in one software technology. We comment on this choice in the following sections. We will call the parts of the workflow running in parallel as the *processes*.

Our workflow architecture is depicted in Fig.6.7. Before the optimization process starts, the STM data is used to generate vessel arrival scenarios (denoted as steps A and B in Fig.6.7). The workflow continues as a SLURM job starting with a hill climber optimizer written in Python (`HC.py`). This piece of code reads quay length Q and admissible berth lengths B, generates quay partitions (`layout.txt`) and calls `Evaluate-1-layout.sh` script executed as a SLURM job (steps 1, 2 in Fig.6.7). This script merges scenarios with layout, thus creating BAP instances, and invokes `Evaluate-1-instance.sh` scripts which execute a scheduling code `schedule` for various pairs of BAP instance and scheduling algorithm (steps 3, 4, 5 in Fig.6.7). This part of the workflow may have different implementations: both `Evaluate-1-instance.sh` and `schedule` can be executed sequentially or in parallel. For `Evaluate-1-instance.sh` we assumed two options: either all the BAP instances are processed sequentially, or in parallel as SLURM jobs started by `sbatch` calls (10 parallel jobs). The `schedule` codes were executed sequentially or in groups of 2, 4, 8 parallel processes by invoking them with ampersand (&). Hence, we will report

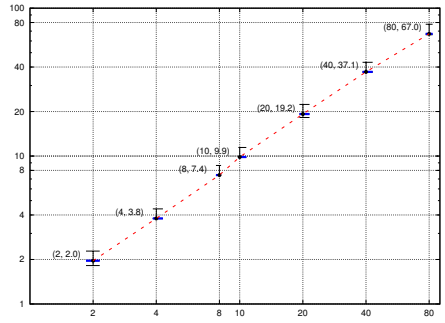


Figure 6.8: Complete workflow speedup vs m of the workflow for traffic intensity $\iota = 500$.

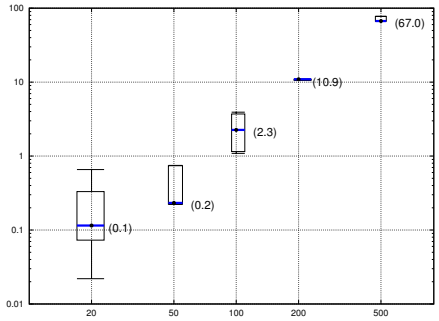


Figure 6.9: Complete workflow speedup vs traffic intensity ι for parallelism $m = 80$.

on the workflow performance with 1, 2, 4, 8, 10, 20, 40, 80 parallel processes. Higher parallelism degrees were also used because in the full enumeration considered in Section 6.4 all partitions were set to run in parallel by use of `sbatch` command. BAP schedules built by `schedule` are stored as text files and statistics on the schedule quality for the tested quay layout are calculated (steps 6,7). These results are fed back to the hill climber code as layout quality $F(K)$ (step 8). For the single-pass algorithms like BBF, ILP1, ILP2 the workflow is executed once for one layout. For Rnd10 it is repeated ten times for ten random layouts. In the case of tabu search the workflow operates in the same way, only the `HC.py` code is substituted.

6.6.2 Time Scalability

The time performance of our workflow run on the SLURM platform was measured as speedup vs number of processes executed in parallel and vs size of the simulated traffic.

The speedup vs number of processes run in parallel is shown in Fig.6.8 for traffic intensity $\iota = 500$ a/w. This is equivalent to a BAP instance with roughly 26000 vessels. In this figure speedup minimum, median and maximum are shown. The dotted line connects medians and at each median point a pair of the number of parallel processes and the obtained speedup are shown for clarity. It can be verified that the workflow scales well in this case because speedup grows from 2 to 67 for two to eighty parallel processes, respectively. Such a behavior could be intuitively expected because the crucial part of the optimization, which is the evaluation of a berth layout, consists of the mutually independent evaluations of the traffic scenarios by a set of

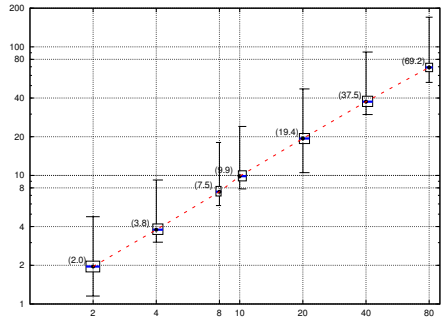


Figure 6.10: Speedup of the evaluation step vs m of the workflow for size $\iota = 500$ a/w.

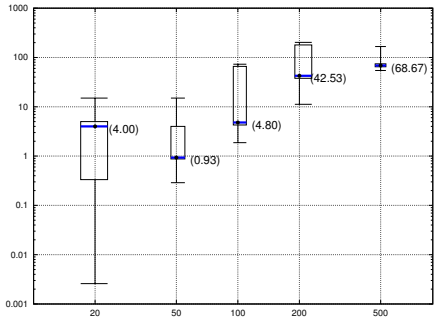


Figure 6.11: Speedup of the evaluation step vs traffic intensity ι for parallelism $m = 80$.

algorithms. Thus, the linear speedup is expected under optimistic assumptions. Unfortunately, the overall picture is not as positive as outlined above. The size of the simulation has a great impact on the performance. Precisely, the intensity of the vessel traffic ι determines evaluation complexity and consequently the relationship between platform-incurred delays and the useful simulation time. This ratio is large (simulations last longer than platform overheads) for high traffic intensity of $\iota = 500$ a/w and hence the speedup is nearly linear as shown in Fig.6.8. For smaller arrival intensities, this ratio is not as advantageous as shown in Fig.6.9. In the figure speedup quartiles are shown vs ship arrival intensities ι. Note that for 80 parallel processes, also speedup should be close to 80 in an application that is scaling well. Unfortunately, this is not the case here. It can be verified that for $\iota \leq 50$ a/w there is no advantage in workflow parallelism. With growing intensity, and hence simulation time, the advantage of parallelism finally emerges. Note that for $m = 80$ parallel processes, our workflow made 10 calls to `sbatch`, sometimes resulting in a significant wait until starting the simulation. This wait time depends on the actual user applications load and the number of active nodes in the computing cluster. It can be guessed that the owners and operators of the SLURM cluster had different objectives (e.g. economizing on energy) than the users, which resulted in a slow reaction to sudden load changes. In our case it had effect in long platform-related delays and low speedup for small arrival intensities visible in Fig.6.9.

A closer verification of the workflow part dedicated to the evaluation of the layout is shown in Fig.6.10 and Fig.6.11. This part of the workflow comprises steps 3–6 shown in Fig.6.7. Intu-

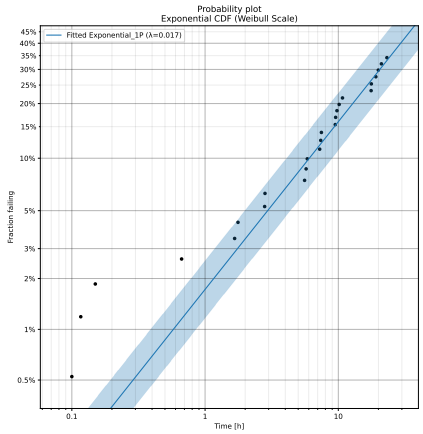


Figure 6.12: CDF of failures in time with 95% confidence bounds.

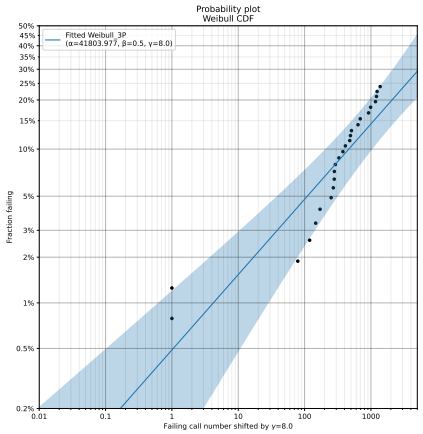


Figure 6.13: CDF of failures over successive SLURM calls with 95% confidence bounds.

itively, this part of the computation should be parallelizable very well because all the evaluations of the quay partition on the instance-algorithm pairs are mutually independent. In the two figures, quartiles of speedup are shown vs parallelism degree (Fig.6.10) and arrival intensity ι (Fig.6.11). For large simulation times, the scalability is good (Fig.6.10). Although the speedups in Fig.6.11 are better for this intensive part of the simulation than for the whole workflow shown in Fig.6.9, parallelism is hardly profitable for $\iota \leq 50$.

6.6.3 Reliability

The SLURM installation that was at our disposal appeared surprisingly prone to failure. To the extent we were able to verify it, the failures originated from the platform because the failures were transient. The failure had two forms. In the first one, the `squeue` revealed a status *"launch failed requeued held"*. It can be found [29, 77] that it probably could have been attributed to some minor and transient SLURM misconfiguration. In such cases `scontrol release <jobid>` command allowed to continue job execution. However, we considered it a failure because the workflow stopped progressing. In the other failure type, a job was called using `sbatch`, it appeared as running in `squeue`, but actually nothing executed and the `slurm-<jobid>.out` file was not created.

Table 6.12: Parameters of the best fitting failure distributions

type of failure	distribution	parameters
on-line time	exponential	$\lambda = 0.0173927$
call number	Weibull	$\alpha = 41804.0$, $\beta = 0.5$, $\gamma = 8$

In the following discussion, we will refer to two models of failure. In the first one, referred to as *online time failure* (OLTF), a failure in SLURM execution happens with the progress of time. Hence, long-running applications accumulate platform failures. This model applies to a whole workflow, or equivalently, processing one instance of a SQPP. In the second model, which will be referred to as *call number failure* (CNF), it is assumed that a failure appears after a certain number of `sbatch` calls. Consequently, applications accumulate chances of SLURM platform failure with each new `sbatch` call. This model applies to `sbatch` calls to SLURM.

The results presented below use the idea of data censoring [101]. The executions that successfully completed are considered right-censored. That is, they could potentially fail in the latter moment, however, the observation of the computation process finished and the failure moment rests in some unknown future time, i.e., to the right of collected observations. In our observations, we have 26 complete data units, that is, 26 whole workflow failures and 189 right-censored data units, i.e., successful workflow executions. In the OLTF model, 26 workflow failure times and 189 successful completion times are used as the right-censored data. Hours are the time units used in the OLTF model. Similarly, for the CNF model, we have 26 failures with the successive number of the `sbatch` call that failed and 189 numbers of `sbatch` calls accumulated in the whole workflow executions that ended correctly (the right-censored data).

The above data were processed with the Python Reliability library [68]. The best-fitting distribution parameters are given in Tab.6.12. These distributions were chosen from the set of exponential, gamma, Weibull, lognormal, loglogistic, normal and Gumbel distributions [68]. The quality of fit was assessed on the basis of the Bayesian Information Criterion (BIC) [100]. Exponential distribution fit best in the OLTF model. For the Weibull distribution that fit best the CNF model of failures, parameter $\alpha = 41803.997$ is often referred to as scale, $\beta = 0.5$ is also called shape, and $\gamma = 8$ is location shift. That is, the fit distribution is shifted to the right by 8 failures. The distributions are visualized as cumulative density functions (CDFs) in Fig.6.12 and Fig.6.13. The results for OLTF can be interpreted in the following way: On average a failure happens after $1/\lambda = 57.5$ hours. For the assumed 24-hour runtime limit the probability

of failure is 0.341. For the CNF model, referring to the number of `sbatch` calls at the assumed limit of 401×10 calls, the probability of failure is 0.341. Possibly, a single-technology monolithic software could allow to avoid the above reliability issues.

6.7 SQPP Summary

In this chapter we considered the quay partitioning problem in a stochastic setting (SQPP). Various research paths presented in the earlier chapters of this thesis were merged here. Ship traffic models introduced in Chapter 5 were elements of SQPP definition. An important component in solving the SQPP is the evaluation of quay partitions. To evaluate the partitions we used a purposefully designed portfolio of algorithms selected in Chapter 4 for short runtime and good quality solutions at the chosen time limits. The nature of SQPP solution features was extensively analyzed on the examples of Le Havre and Shanghai traffic models. It appears that at low traffic intensity most of the quay partitions are almost equally good. Only under intensive traffic, can the alternative solutions to SQPP show their superiority and the more intensive the traffic is, the more different the partitions performances are. Under high traffic, solutions that are good with respect to short waiting time are also good at the dispersion of waiting time. Moreover, good partitions depend on the traffic features. For example, solutions with many long berths are good for intensive large vessel traffic which corresponds with the DQPP results under congestion (Section 3.6.2). Solutions based on building berths of equal lengths or as many big berths as possible are generally not the best, yet for specific STMs can provide satisfactory solutions. We proposed a set of algorithms to construct a quay partition. Brute force enumeration of all possible quay partitions and evaluating them is to a limited degree doable for contemporary terminals. A number of methods with lower complexity were proposed, with the goal of applying them in future terminal designs allowing many more berths. It turned out that these methods can be competitive in various domains of application. Tabu search offers good solution quality, but tends to be computationally demanding. In such situation partitions built by ILP1, ILP2 can be used. We used the opportunity of solving SQPP to shed some light on technical aspects of the optimization workflow. On the one hand, our workflow time efficiency depends on the simulation size and large simulations offer better scalability. On the other hand, long-running applications using many machines may encounter platform reliability issues.

Chapter 7

Conclusions and Final Remarks

This dissertation has explored several aspects of combinatorial optimization, vessel traffic modeling interconnected on the grounds of port logistics, focusing on the Quay Partitioning Problem (QPP), which is critical for efficient vessel servicing and resource allocation in maritime container terminals.

The research began with the Discrete Quay Partitioning Problem, presenting two mixed integer linear programming (MIP) methods partitioning a quay for a single known vessel arrival scenario. Solution feature analysis demonstrated, among the others, that the 2in1 hybrid vessel berthing method outperforms the 1in1 method in terms of robustness and space utilization. The 2in1 hybrid layout, which allows two vessels to share a berth, has been shown to use restricted set of berth lengths better than the classic discrete berthing. Consequently, 2in1 layout results in reducing idle time and increasing throughput. This finding laid the foundation for further analysis of hybrid 2in1 layout in the subsequent sections of the dissertation.

Evaluation of quay partition by simulation on a realistic time horizon requires selecting specialized methods solving very large instances of Berth Allocation Problem (BAP). This fact landed this research in the realm of the Algorithm Selection Problem (ASP). While addressing the ASP, a novel approach for selecting an optimal portfolio of algorithms for the BAP was proposed. The ASP framework enhances the robustness and adaptability of the proposed solutions to the required computation time budget, providing a systematic way to choose the best-performing algorithms for different scenarios. No algorithm is excluded from the portfolio

as long as there is an instance and time limit under which this algorithm provides the best solution. Thus, the portfolio evolves with the runtime limits. This approach ensures that the selected algorithms are well-suited to the specific characteristics of the problem.

The Ship Traffic Model (STM) was introduced to create realistic test instances for QPP. By analyzing historical ship traffic data from selected ports, the STM accurately reflects real-world conditions and dependencies. A detailed consideration of individual port traffic characteristics and the arriving ship attributes interactions ensures that the test scenarios are robust and representative. The STM is instrumental in validating the proposed quay partitioning strategies under realistic conditions, highlighting the importance of data-driven modeling in port logistics research. As an additional contribution, a method of comparing ports emerged. It was found that each port has unique elements in its STM which are consequences of the location (mainly), mix of clients and shipped goods.

Finally, the dissertation explored the Stochastic Quay Partitioning Problem (SQPP) in which the unknown future vessel traffic is represented by arrival scenarios generated from the earlier STMs and the partition performance is evaluated by the tailored portfolio of algorithms for BAP. A set of algorithms was designed to partition the quay in such stochastic setting. It was discovered that partitioning the quay into equal length berths is not the best option. It was established already in the deterministic case that such partitions can be arbitrarily bad. In the stochastic case they are not among the worst, with respect to the mean waiting time, but are unwieldy which emanates in large dispersion of performance evaluations. Furthermore, large ships dominate in defining good partitions. It was confirmed both in the discrete (DQPP) and in the stochastic case (SQPP). This part of the research underscores the importance of flexibility and resilience in quay partitioning strategies, demonstrating that stochastic optimization techniques can significantly improve performance in uncertain environments. The results obtained for DQPP and SQPP demonstrated that quay partitioning has great impact on container terminal performance. Low values of vessel waiting times can only be obtained by selected quay partitions. Consequently, we consider that the research hypothesis posed in Section 1.2 has been confirmed.

This study opens avenues to new research directions. The QPP deserves developing dedicated algorithms such as, e.g., metaheuristics both in the deterministic and in the stochastic setting. The methodology of ASP developed to choose BAP algorithm portfolios can be applied in other

combinatorial optimization problems. We hope that the results and methods presented in this thesis will help both researchers and practitioners designing container terminals of the future.

Appendix A

Streszczenie w języku polskim

Globalna żegluga kontenerowa jest jednym z kluczowych i najtrudniejszych sektorów gospodarki. Typowe dla niej problemy można odwzorować na szeroko badane zagadnienia optymalizacji kombinatorycznej i badań operacyjnych. Podczas projektowania nowego morskiego terminala kontenerowego lub przebudowy istniejącego, kluczową decyzją jest podział linii brzegowej (ang. quay) na nabrzeża (ang. berths). Technicznie, układ nabrzeży terminala to podział jednego długiego odcinka (linii brzegowej) na wiele mniejszych (nabrzeża). Rozwiązanie problemu QPP wymaga określenia liczby nabrzeży i ich długości w taki sposób, aby czas oczekiwania na obsługę przybywających kontenerowców był zminimalizowany. W rozprawie doktorskiej stawiana jest następująca hipoteza: *Wydajność morskiego terminala kontenerowego można poprawić poprzez optymalizację podziału linii brzegowej na nabrzeża.*

Pierwszy z rozważanych w pracy problemów, deterministyczny problem podziału linii brzegowej (ang. Deterministic Quay Partitioning Problem, DQPP), sformułowany został w dwóch wariantach: 2w1 i 1w1, dopuszczających odpowiednio przydział maksymalnie dwóch statków lub tylko jednego statku do poszczególnych nabrzeży. Oba warianty okazały się **NP**-trudne nawet w bardzo restrykcyjnych przypadkach. Dla obu przypadków skonstruowano zadanie mieszanego programowania liniowego (ang. Mixed Integer Linear Programming, MIP). Oceniono skalowalność czasową rozwiązania DQPP za pomocą MIP. Przeanalizowano, m.in. jak rozrzut wielkości statków wpływa na wybrane długości nabrzeży.

W pracy rozważany jest problem wyboru algorytmów (ang. Algorithm Selection Problem,

ASP) do rozwiązania dużych problemów przydziału statków do nabrzeży (ang. Berth Allocation Problem, BAP) z ograniczeniami budżetu czasowego. Indywidualna ocena 72 szybkich heurystyk doprowadziła do wniosku, że pewne algorytmy zachłanne są najlepsze dla krótkich budżetów czasowych, a metaheurystyki są najlepsze dla dużych budżetów czasowych. Zbadano również koncepcję portfolio algorytmów do rozwiązania BAP. Badanie ewolucji portfolio w czasie nie tylko potwierdziło wcześniejszą jakościową intuicję co do stosowalności konkretnych heurystyk, ale także dostarczyło ilościowego sposobu przechodzenia z pierwszego zestawu algorytmów do drugiego. Wyniki przedstawione w tym rozdziale mogą być interesujące z punktu widzenia metodologicznego, ponieważ zaproponowano niestandardowe podejście do oceny heurystyk umożliwiające kompromis między długością czasu wykonania a jakością rozwiązań.

W pracy analizowane są wzorce ruchu kontenerowców w ośmiu portach na świecie. Ma to na celu opracowanie modeli ruchu (ang. Ship Traffic Model, STM) do optymalizacji i symulacji logistyki portowej. Modele te zostały wykorzystane w rozdziale 6. Opracowano bardziej zaawansowane i szczegółowe modele niż istniejące we wcześniejszej literaturze. Modele uwzględniają zależności między rozmiarami statków, czasami przetwarzania (obsługi) i czasami przybycia. Każda klasa wielkości kontenerowców ma swój własny model czasu przetwarzania i czasu powrot w postaci reprezentacji statystycznej. Szczególną uwagę zwraca się na ich rozkłady. Brane są pod uwagę również statki aperiodyczne.

W pracy rozważano problem podziału linii brzegowej w wariancie stochastycznym (ang. Stochastic Quay Partitioning Problem, SQPP). Różne ścieżki badawcze przedstawione we wcześniejszych rozdziałach tej pracy zostały tutaj połączone. Modele ruchu statków wprowadzone w rozdziale 5 były elementami definicji SQPP. Aby ocenić podziały linii brzegowej, użyto celowo zaprojektowanego portfolio algorytmów wybranych w rozdziale 4 dla zadanego krótkiego czasu symulacji i dobrej jakości rozwiązań w wybranych granicach czasowych. Rozwiązania oparte na budowaniu nabrzeży o równej długości lub jak największej liczby dużych nabrzeży nie są na ogół najlepsze, jednak dla niektórych STM mogą zapewnić zadowalające rozwiązania. Wyliczenie metodą brutalnej siły wszystkich możliwych podziałów nabrzeża i ich ocena są w ograniczonym stopniu wykonalne dla współczesnych terminali, takich jak Hawr lub Szanghaj. Zaproponowano szereg metod o mniejszej złożoności, których celem jest zastosowanie w przyszłych projektach terminali umożliwiających budowę znacznie większej liczby nabrzeży.

Appendix B

Summary

Global container shipping is one of the key and most challenging sectors of the economy. The typical problems can be successfully mapped to the widely studied problems of combinatorial optimization and operations research. When designing a new maritime container terminal or rebuilding an existing one, a key decision is the division of the quay into berths. Technically, the layout of the terminal quay is the division of one long segment (the quay) into many smaller ones (the berths). The solution requires determining the number of berths and their lengths in such a way that the waiting time of the arriving container ships is minimized. In this doctoral dissertation, the following hypothesis is defended: *The efficiency of a maritime container terminal can be improved by optimizing the partitioning of the quay into berths.*

The first considered problem, the Deterministic Quay Partitioning Problem (DQPP), is formulated in two variants: 2in1 and 1in1, allowing for the allocation of at most two ships or only one ship to individual berths, respectively. Both variants turned out to be **NP**-hard even in very restrictive cases. For both cases, a mixed integer linear program (MIP) was constructed. The time scalability of the DQPP solution using MIP was assessed. Among the others, the impact of the dispersion of ship sizes on the selected berths lengths was analyzed.

This thesis considers the problem of selecting algorithms (Algorithm Selection Problem, ASP) for solving large Berth Allocation Problems (BAP) with time budget constraints. The individual evaluation of 72 fast heuristics led to the conclusion that certain greedy algorithms are best for short time budgets, while metaheuristics are best for large time budgets. The concept

of a portfolio of algorithms for solving BAP was also investigated. The study of the evolution of the portfolio in time not only confirmed the previous qualitative intuition about the use of specific heuristics, but also provided a quantitative way of moving from the first set of algorithms to the second. The results presented in this Chapter can be interesting from a methodological point of view, as a non-standard approach to the evaluation of heuristics is proposed, including a trade-off between the execution time and the solution quality.

Traffic patterns of container ships in eight ports around the world are analyzed in this thesis. The aim is to develop traffic models (Ship Traffic Models, STMs) for the optimization and simulation of port logistics. These models were used in Chapter 6. More advanced and detailed models than those in the previous literature were developed. The models take into account the relationships between ship sizes, processing (handling) times and return times. Each vessel size class has its own statistical processing time model and retrn time model. Particular attention is paid to the distributions of these times. Aperiodic ships are also considered.

In the thesis Stochastic Quay Partitioning Problem (SQPP) was considered in Chapter 6. Different research paths presented in the previous chapters were combined here. The ship traffic models introduced in Chapter 5 were elements of the SQPP definition. To evaluate the quay partitions, a purposefully designed portfolio of algorithms selected in Chapter 4 was used for the given short simulation time budget and good quality of solutions. It turned out that solutions based on building equal length berths or as many large berths as possible are generally not the best, but for specific STMs they may provide satisfactory solutions. Brute force enumeration of all possible quay divisions and their evaluation, are to a limited extend feasible for modern terminals such as Le Havre or Shanghai. A number of less complex algorithms have been proposed with the aim of designing future terminals significantly more berths.

Appendix C

Algorithm Portfolios for Large Berth Allocation Problems

C.1 Introduction

Algorithm Selection Problem (ASP) consists in selecting a method most suitable to solve an instance of a certain problem. Algorithm portfolios are one of the ways of addressing the ASP. Algorithm portfolios are sets of algorithms which can be applied collectively to solve a certain problem. The algorithms can be, e.g., executed sequentially or in parallel, on a given instance of the considered problem and the best solution is chosen. Here we study algorithm portfolios for large instances of Berth Allocation Problem (BAP) [7]. The BAP must be solved as sub-problem in simulations of a port, and hence the execution time T of the portfolio is a key constraint.

The BAP considered in this paper is formulated as follows. A set of m berths of lengths λ_i, for $i = 1, \ldots, m$, is given. There are n vessels defined by: arrival times r_j, lengths L_j, processing (unloading and loading) times p_j, and importance w_j, for $j = 1, \ldots, n$. Vessel j can be moored at berth i only if $L_j \leq \lambda_i$. The quay is divided into discrete berths and each berth can accommodate at most two ships at the same time. The minimized objective is the mean weighted flow time $MWFT = \sum_{j=1}^{n}(c_j - r_j)w_j / \sum_{j=1}^{n} w_j$, where c_j is completion time of handling ship j.

Let $\mathcal{I}'$ be a set of training instances and let $\mathcal{A}$ be a set of algorithms. For each pair $(a, I) : a \in$

$\mathcal{A}, I \in \mathcal{I}'$ a sequence of pairs $(t_{aI}^1, q_{aI}^1), (t_{aI}^2, q_{aI}^2), \ldots, (t_{aI}^{K_{aI}}, q_{aI}^{K_{aI}})$ representing solutions built by a for I over time is given. The t_{aI}^j are execution times and q_{aI}^j are mean weighted flow times. If some algorithm a does not improve solution quality over time then only one pair (t_{aI}^1, q_{aI}^1) is given. Let $t(a, I, T)$ denote runtime of algorithm a on instance I with the time limit T. Let $y(a, I, T) = 1$ if $\exists j : t_{aI}^j \leq T, q_{aI}^j = q_{\min}(I, T)$, otherwise $y(a, I, T) = 0$.

In the following sections name *cover portfolio* refers to the algorithm portfolios obtained by solving the integer linear program (ILP) covering the test instances with minimum cost wins:

$$\min Cost \tag{C.1}$$

$$\sum_{a \in \mathcal{A}} y(a, I, T) x_a \geq 1 \quad \forall I \in \mathcal{I}' \tag{C.2}$$

$$\sum_{a \in \mathcal{A}} t(a, I, T) x_a \leq Cost \quad \forall I \in \mathcal{I}' \tag{C.3}$$

Name *regret portfolio* will refer to the algorithm portfolios obtained by solving the integer linear program covering the test instances by minimizing the greatest regret under limited computational $Cost$ (*Min-Max Regret (MaxReg)*:

$$\min G \tag{C.4}$$

$$\frac{q(a, I, T)}{q_{\min}(I, T)} u_{aI} \leq G \qquad \forall I \in \mathcal{I}' \quad \forall_{a \in \mathcal{A}} \tag{C.5}$$

$$\sum_{a \in \mathcal{A}} u_{aI} \geq 1 \qquad \forall I \in \mathcal{I}' \tag{C.6}$$

$$x_a \geq u_{aI} \qquad \forall I \in \mathcal{I}' \quad \forall_{a \in \mathcal{A}} \tag{C.7}$$

$$\sum_{a \in \mathcal{A}} t(a, I, T) x_{aI} \leq Cost \qquad \forall I \in \mathcal{I}' \tag{C.8}$$

Set $\mathcal{A}$ of evaluated algorithms has 72 members. In this, 60 algorithms are greedy, 12 are metaheuristics. The algorithms have been described in more detail in [92]. For the training and evaluation purposes set $\mathcal{I}'$ of random instances have been generated as follows: $n \sim U[1, 1000]$, $m \sim U[1, 100]$, $r_j \sim U[0, 1000]$, $p_j \sim U[1, 24]$, $w_j \sim U[1, 1000]$, $L_j, \lambda_i \sim U\{200, 215, 290, 305, 400\}$. By $\sim U[a, b]$ we denote that certain parameter is generated from discrete uniform distribution with integer values in range $[a, b]$. $\sim U\{x, \ldots, y\}$ denotes that the parameter values are chosen with discrete uniform distribution from the set $\{x, \ldots, y\}$. Unless stated to be other-

wise, each configuration of the tests represents a population of 100 instances. In the tests a range of n values has been swept by visiting values $n \in \{2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000\}$ one by one. This means that 100 instances have been generated with $n = 2$ while the remaining parameters have been randomly generated as described above. The set of 100 instances with $n = 2$ have been solved by all the algorithms to evaluate their performance. Next, values $n = 5, 10, \ldots$ have been examined in the similar manner. This set of of instances will be referred to as *random instances* N. In the examination of the impact of m, the tested values were $m \in \{1, 2, 5, 10, 20, 50, 100\}$ while other parameters were generated as described above. These instances will be referred to as *random instances* M.

In the next dataset, referred to as *real* instances, a collection of ship arrival times (r_j), processing times (p_j), lengths L_j, has been obtained from Automatic Identification System (AIS), a global radio vessel tracking system, for 8 ports in 2016. The ports with the number of ships were: Gdańsk – 465 ships, Long Beach – 995, Los Angeles – 1310, Le Havre – 2277, Hamburg – 3273, Rotterdam – 3997, Shanghai – 11197, Singapore – 18413. Quay lengths were obtained from the publicly available port authority data. Overall, the algorithm portfolios have been constructed for the following datasets:

dataset 1 all 1200 random instances N,

dataset 2 100 random instances N with $n = 10000$,

dataset 3 700 all random instances M,

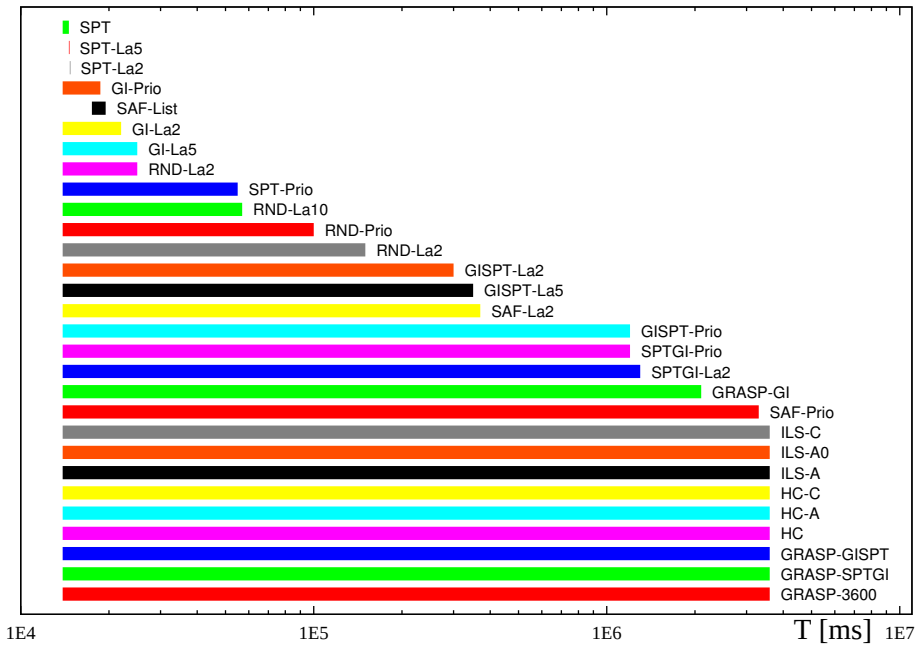
dataset 4 100 random instances M with $m = 2$,

dataset 5 Real instances.

More on the considered problem, its application, and solution will be given in [88].

C.2 Cover algorithm portfolios

a)



b)

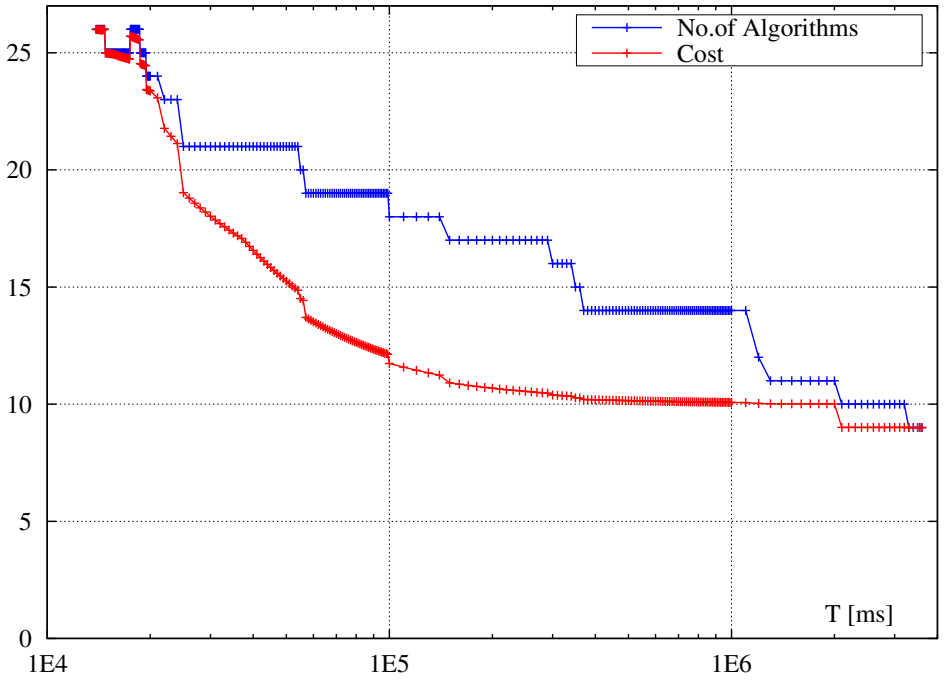
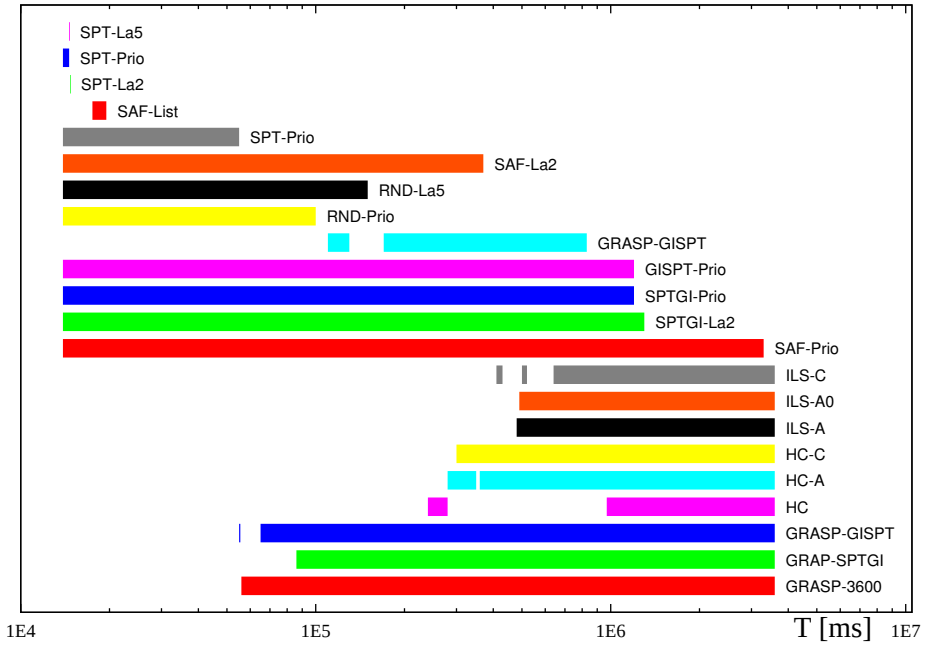


Figure C.1: Cover portfolio built on all random instances N (dataset 1). a) portfolio evolution in time T, b) size and cost of portfolio (in T units), vs T.

a)



b)

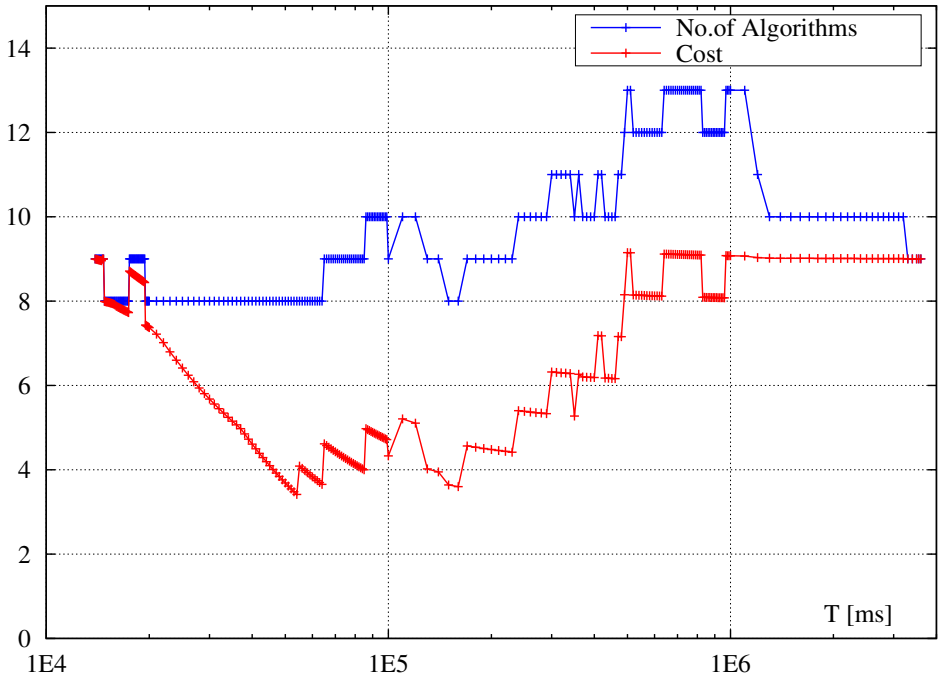
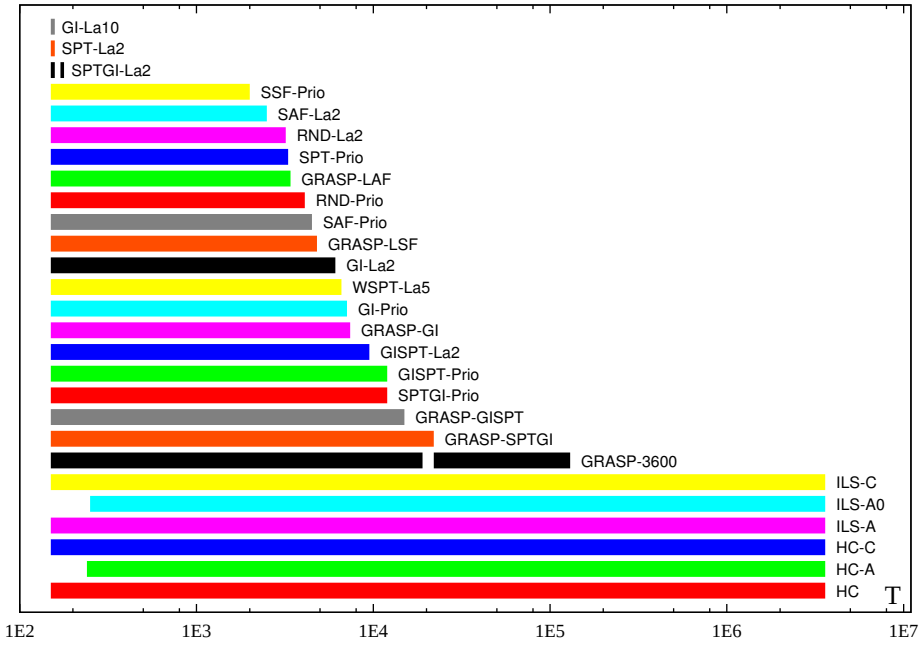


Figure C.2: Cover portfolio built on random instances N for $n = 10000$ (dataset 2). a) portfolio evolution in time T, b) size and cost of portfolio (in T units), vs T.

a)



b)

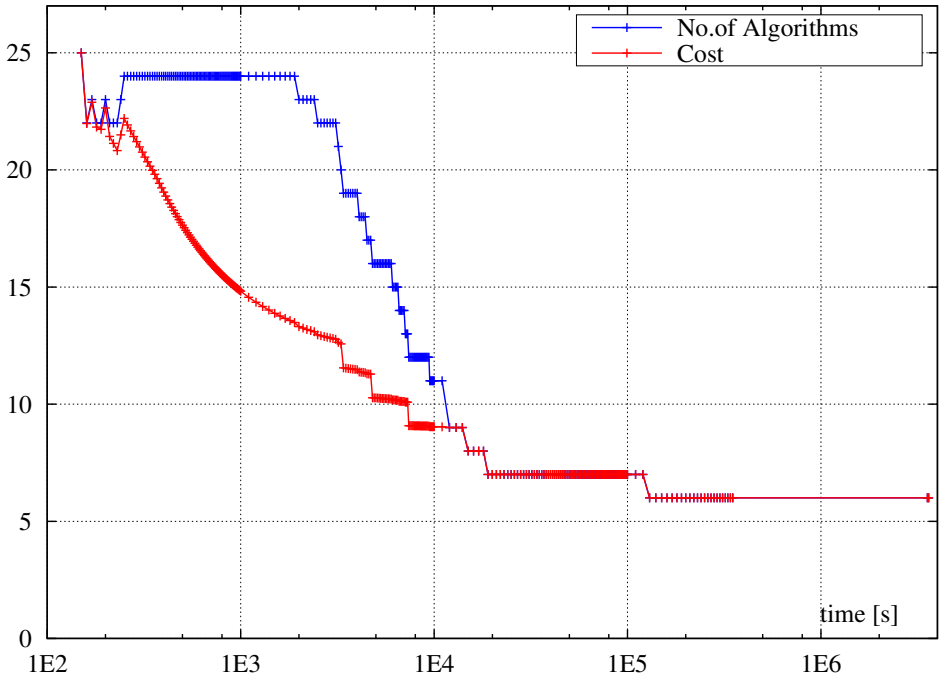
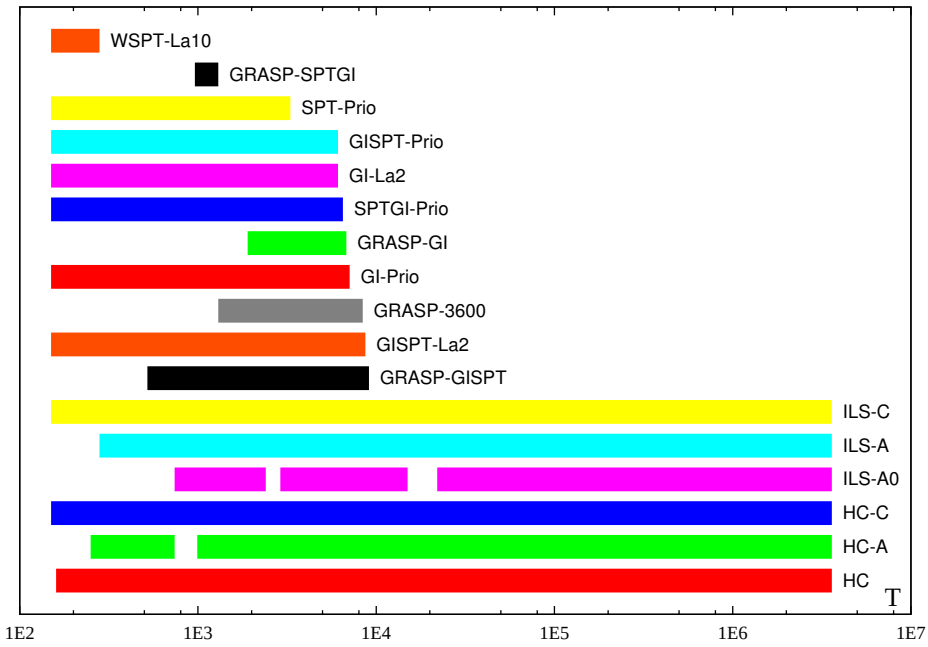


Figure C.3: Cover portfolio built on all random instances M (dataset 3). a) portfolio evolution in time T, b) size and cost of portfolio (in T units), vs T.

a)



b)

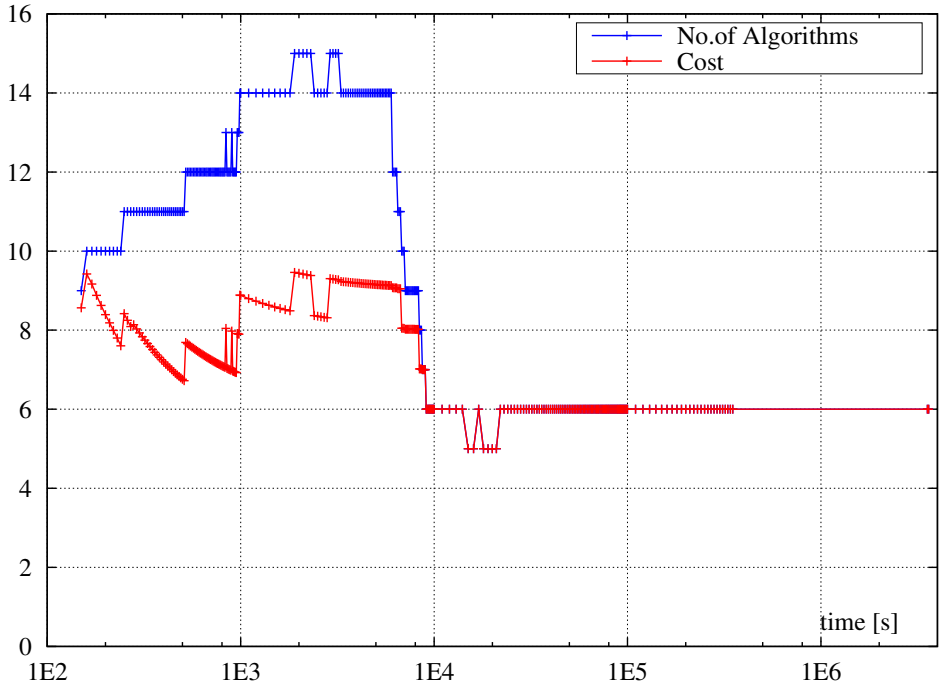
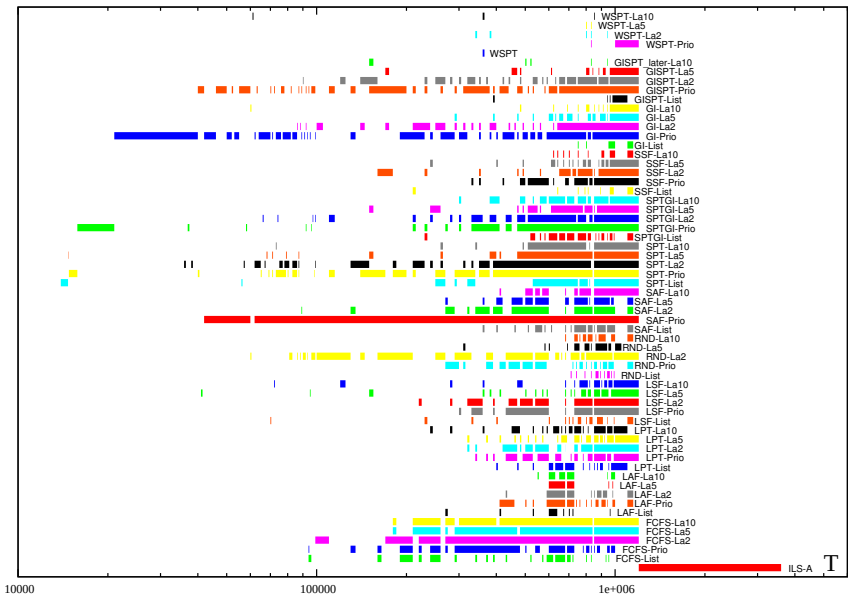


Figure C.4: Cover portfolio built on random instances M with $m = 2$ (dataset 4). a) portfolio evolution in time T, b) size and cost of portfolio (in T units), vs T.

C.3 Regret portfolios for *all* random instances N (dataset 1)

Let us observe that regret portfolios for sufficiently large $Cost$ limit comprise all algorithms (see Figs C.9, C.10, C.15, C.16, C.21, C.22, C.27, C.28).

a)



b)

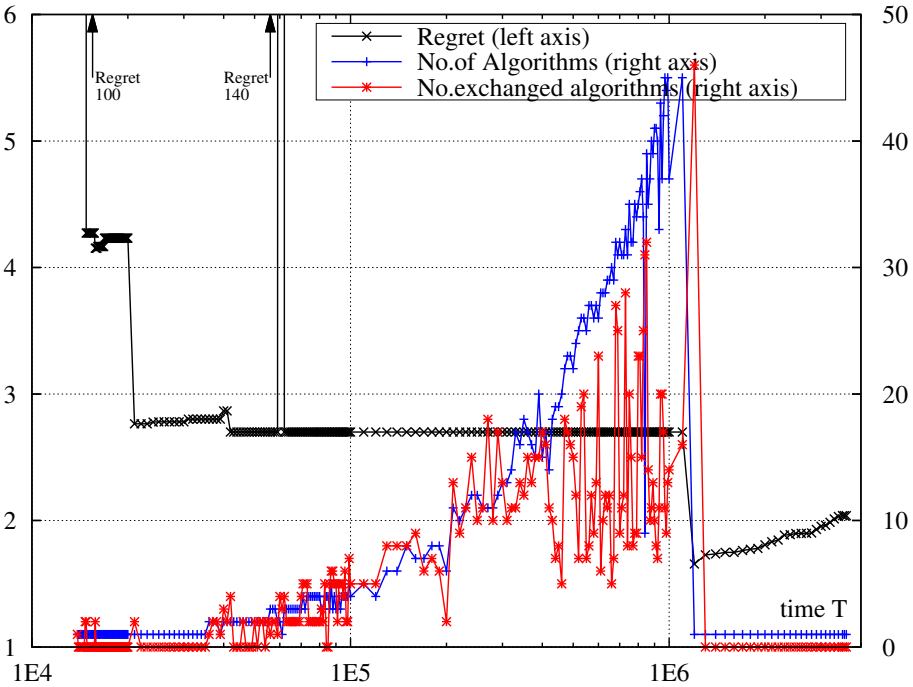
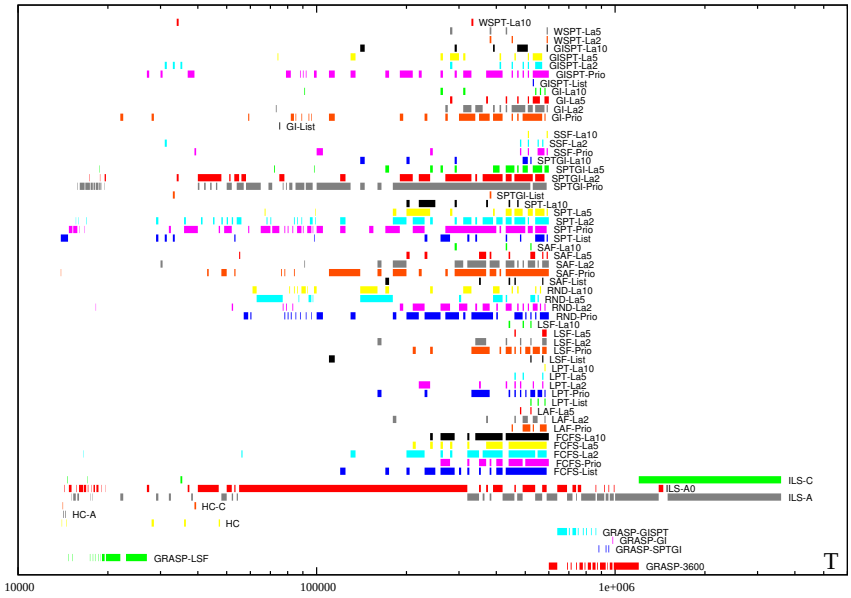


Figure C.5: Algorithm portfolio built on all random instances N, (dataset 1) $Cost = 1T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

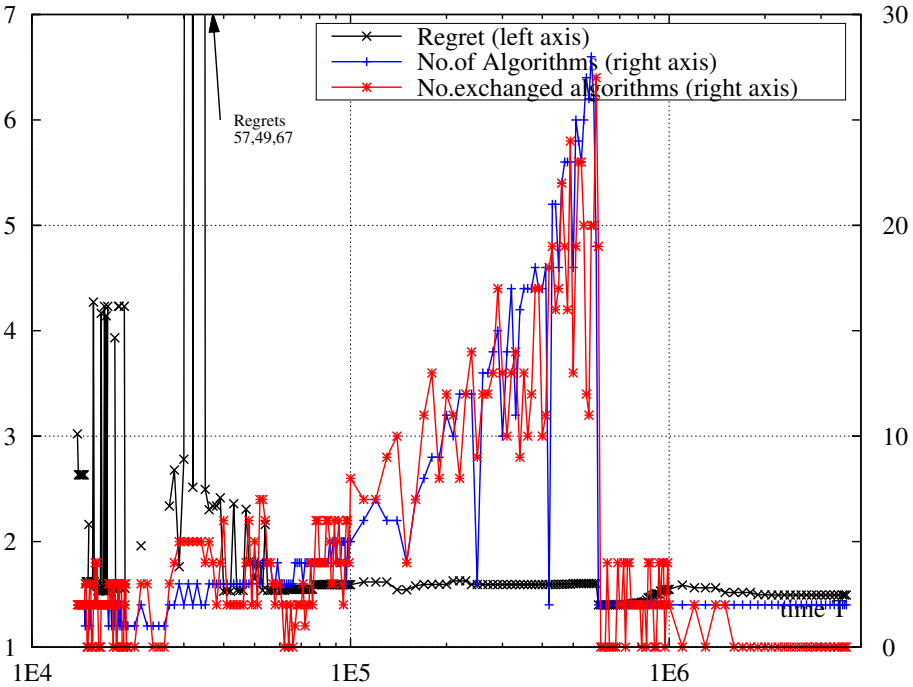
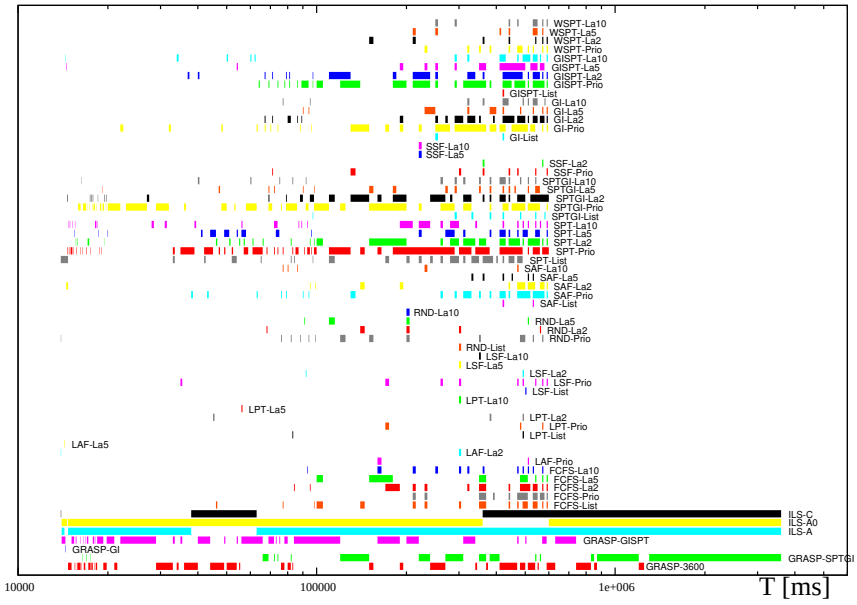


Figure C.6: Algorithm portfolio built on all random instances N, (dataset 1) $Cost = 2T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

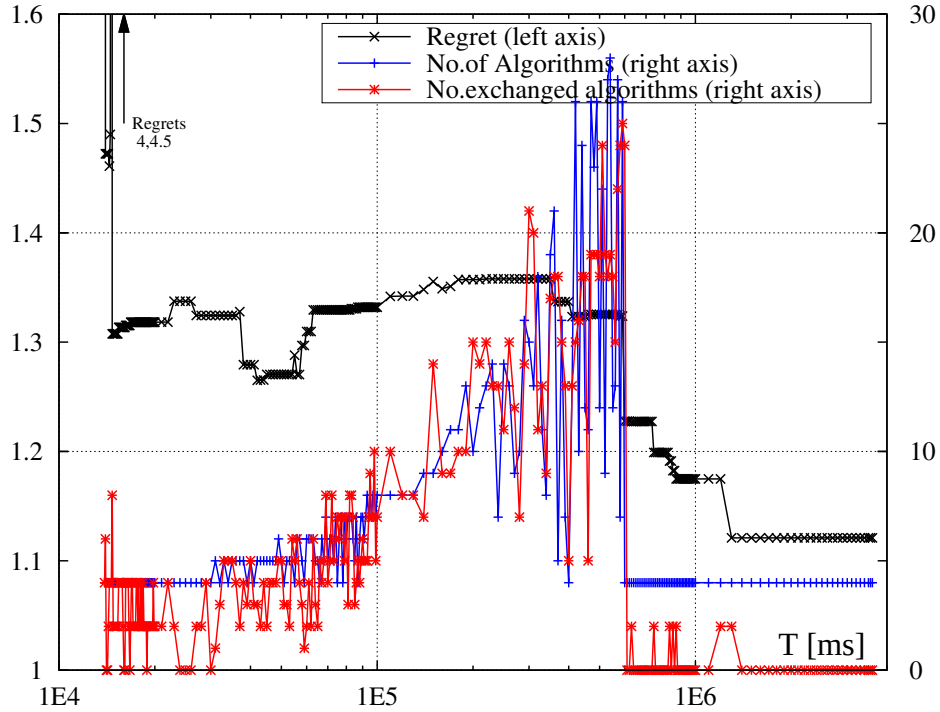
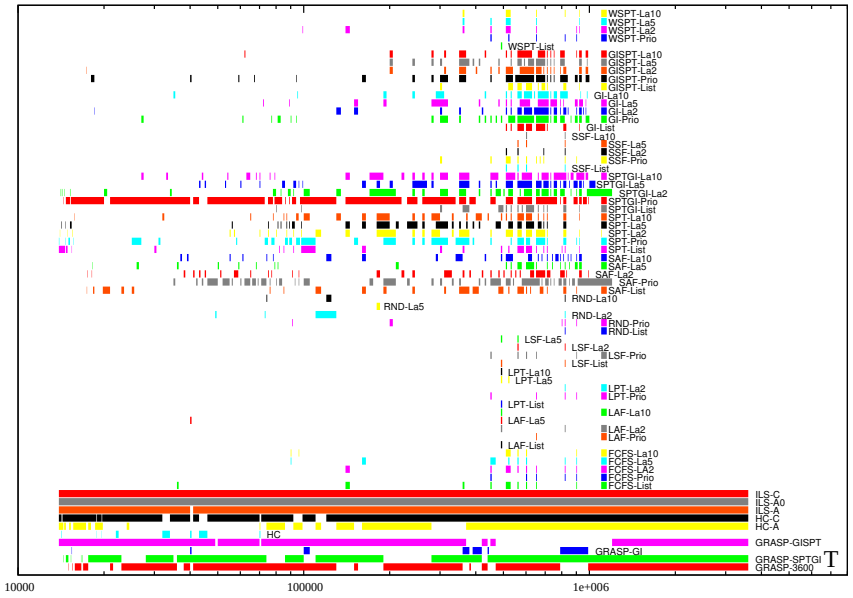


Figure C.7: Algorithm portfolio built on all random instances N, (dataset 1) $Cost = 4T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

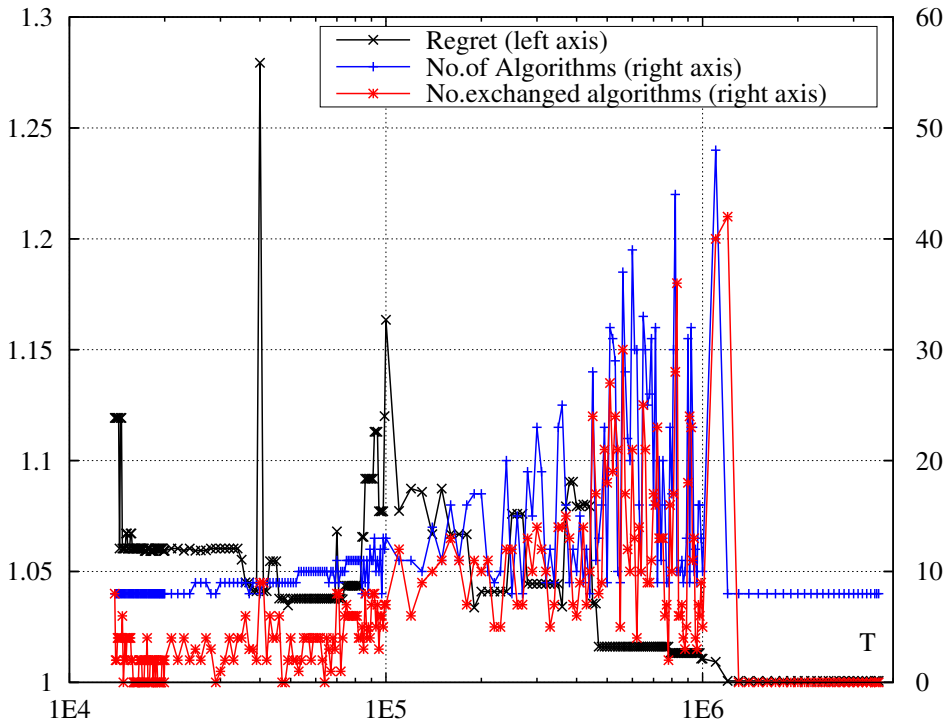
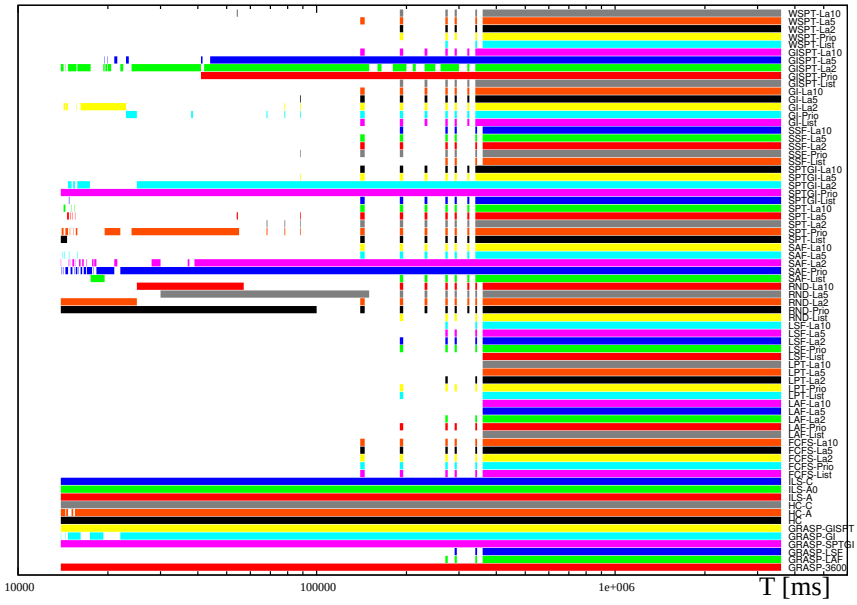


Figure C.8: Algorithm portfolio built on all random instances N, (dataset 1) $Cost = 8T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

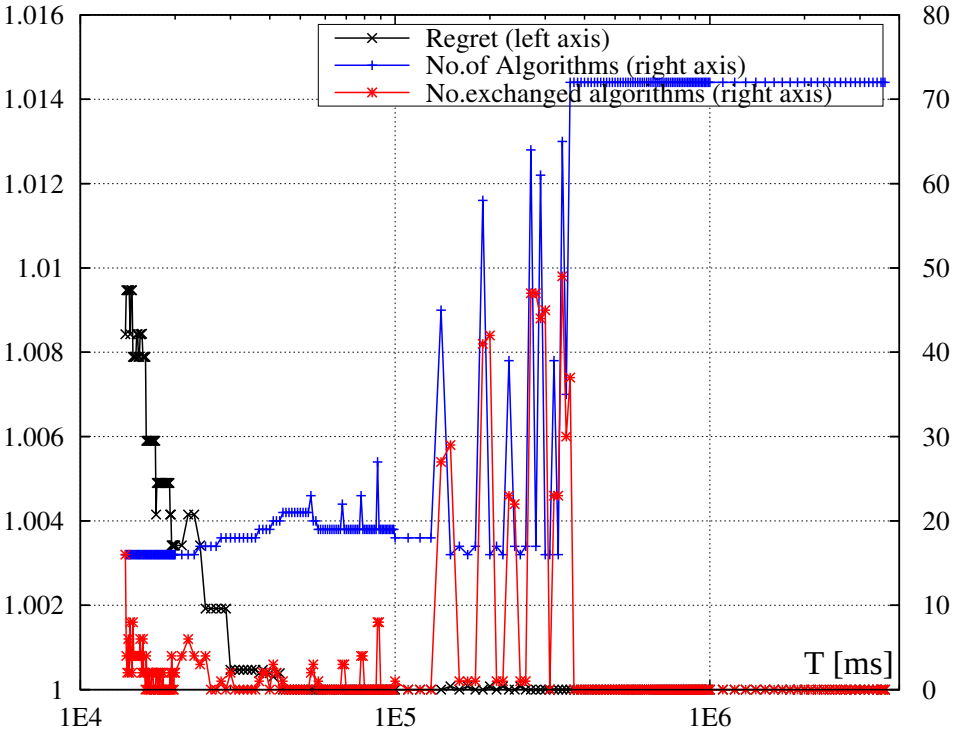
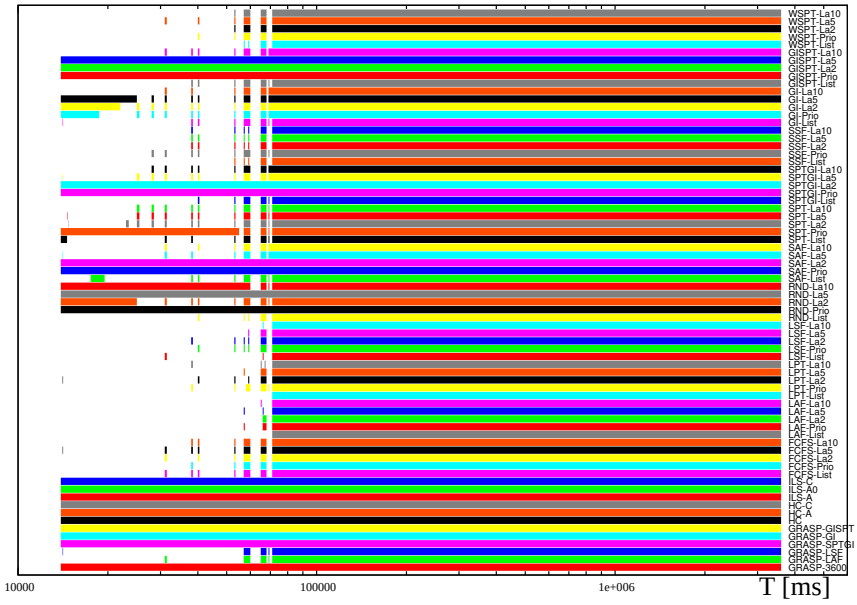


Figure C.9: Algorithm portfolio built on all random instances N, (dataset 1) $Cost = 16T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

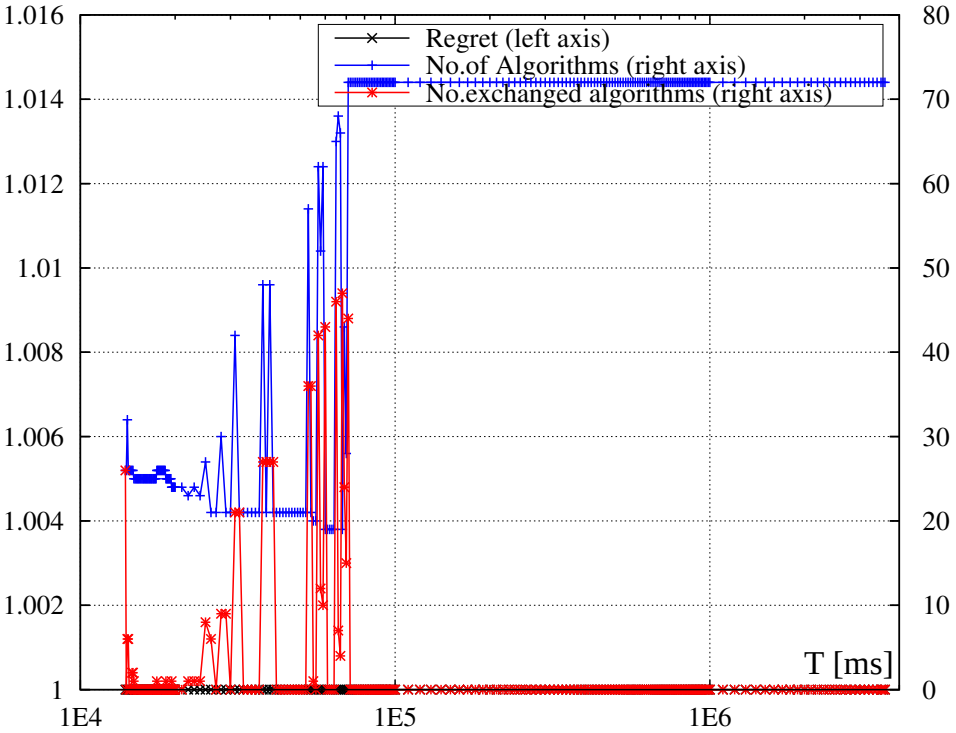
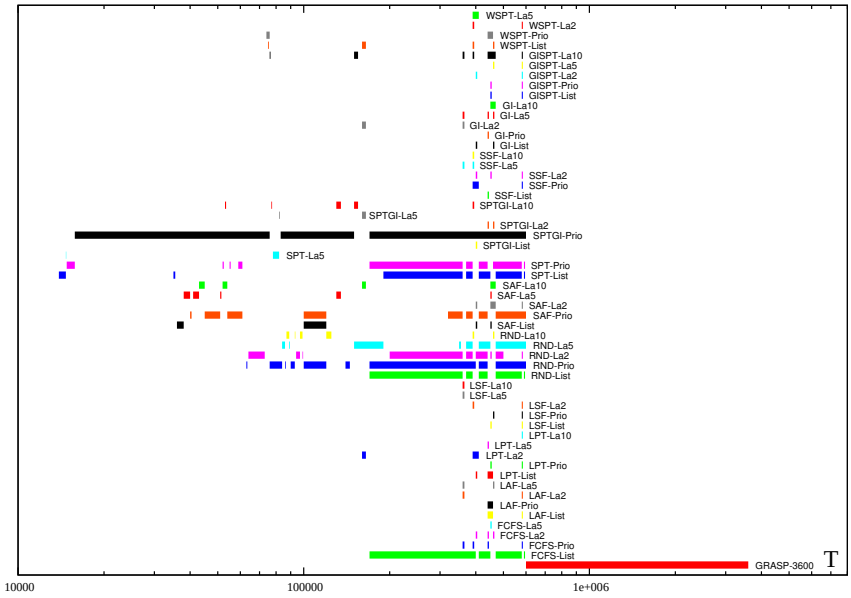


Figure C.10: Algorithm portfolio built on all random instances N, (dataset 1) $Cost = 32T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

C.4 Regret portfolios for *n=10000* random instances N (dataset 2)

a)



b)

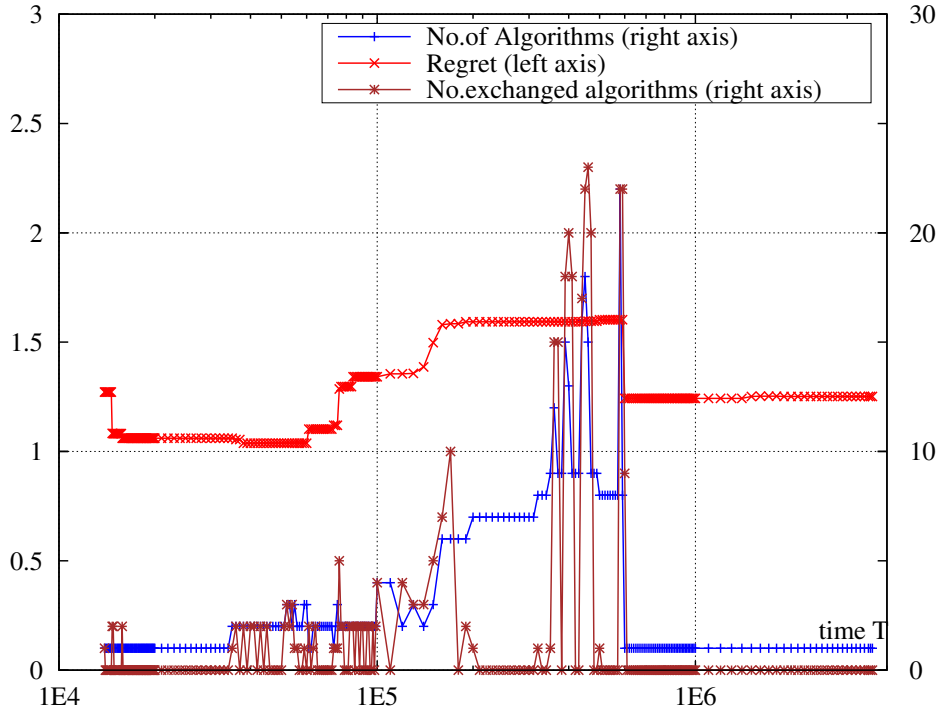
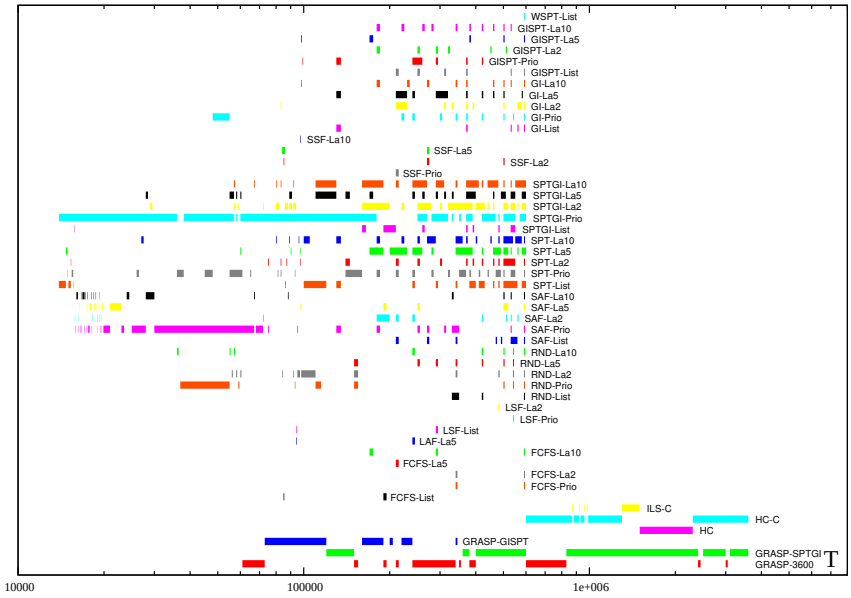


Figure C.11: Algorithm portfolio built on random instances N with $n = 10000$, (dataset 2) $Cost = 1T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

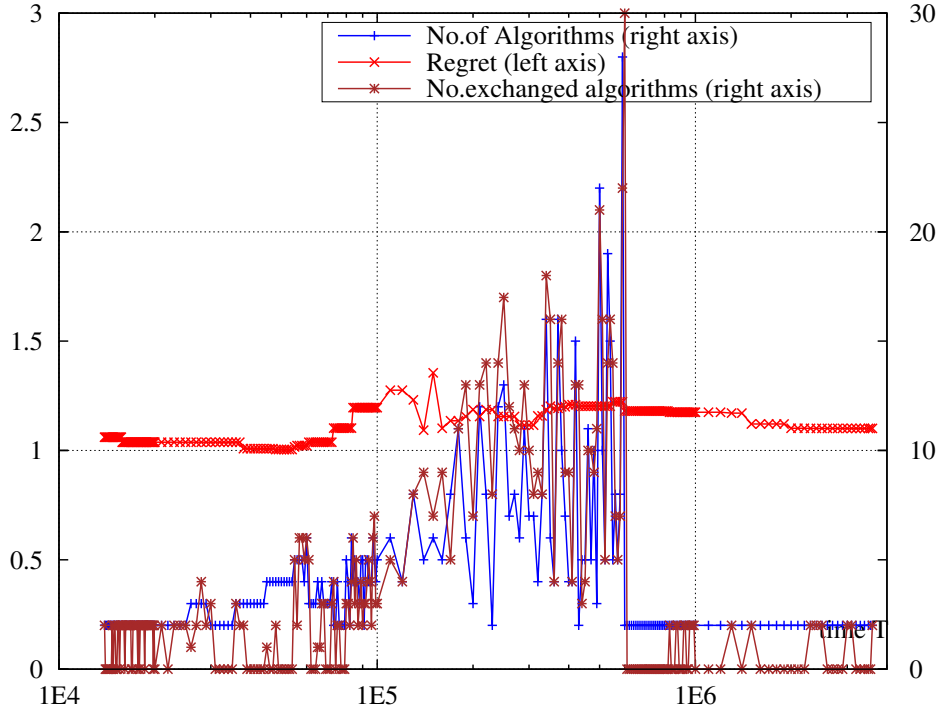
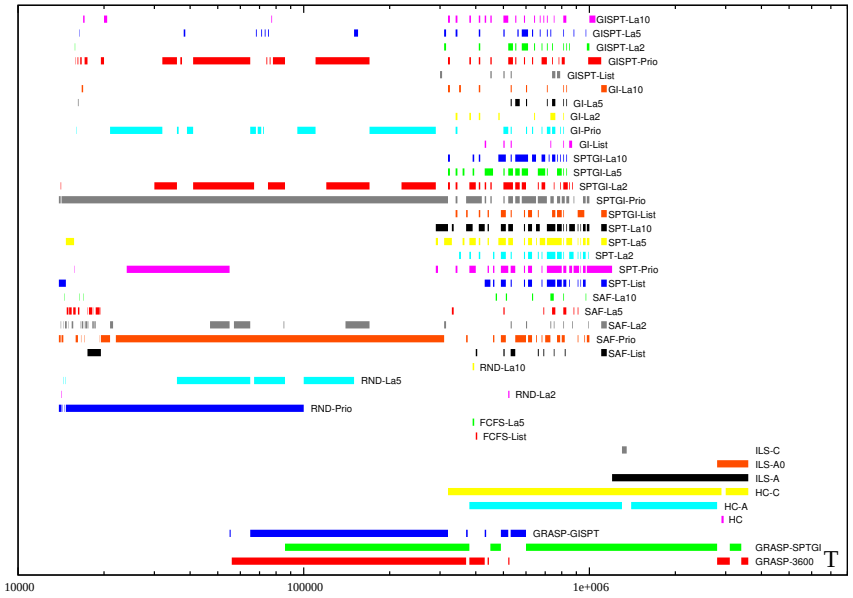


Figure C.12: Algorithm portfolio built on random instances N with $n = 10000$, (dataset 2) $Cost = 2T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

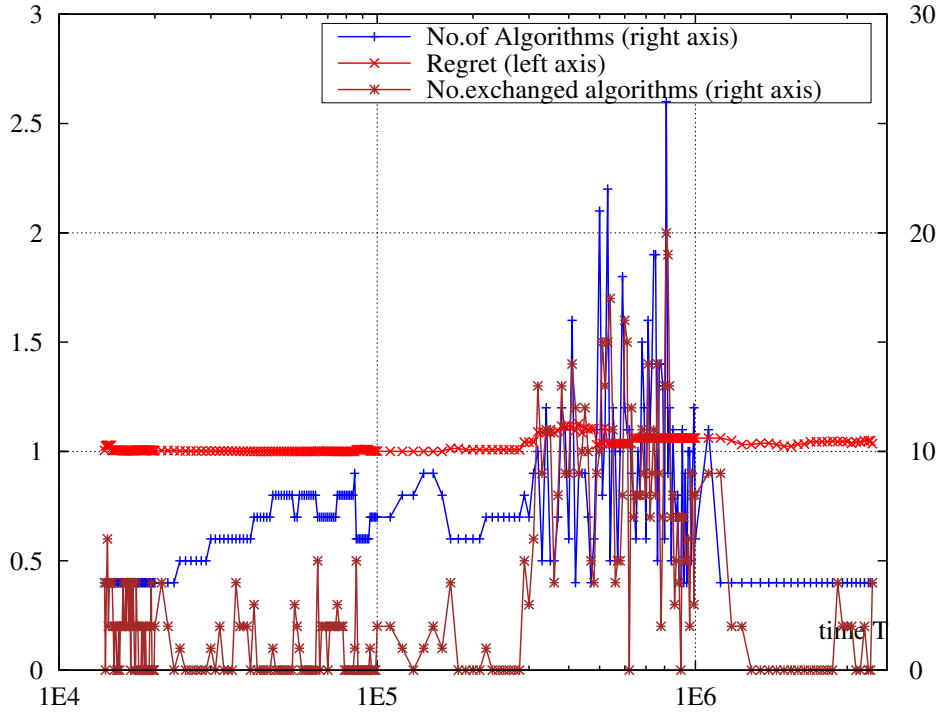
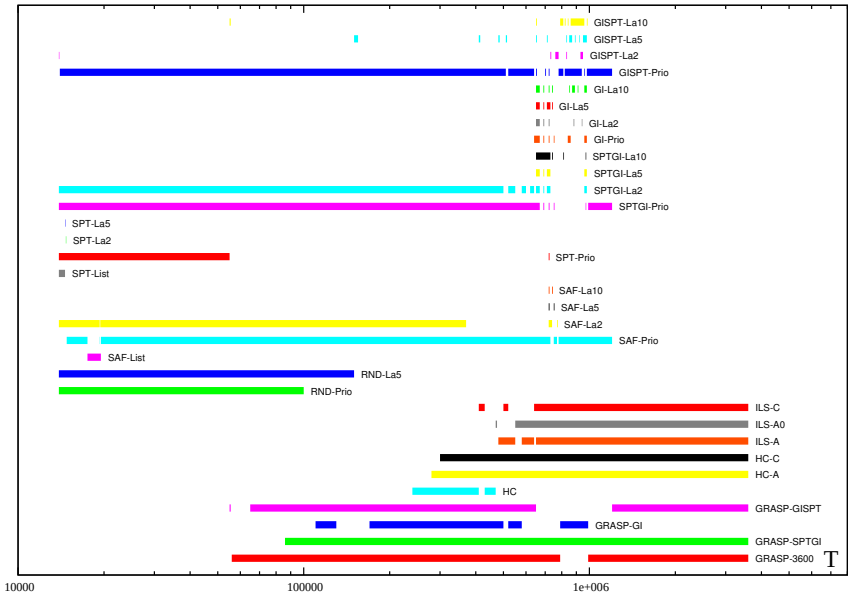


Figure C.13: Algorithm portfolio built on random instances N with $n = 10000$, (dataset 2) $Cost = 4T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

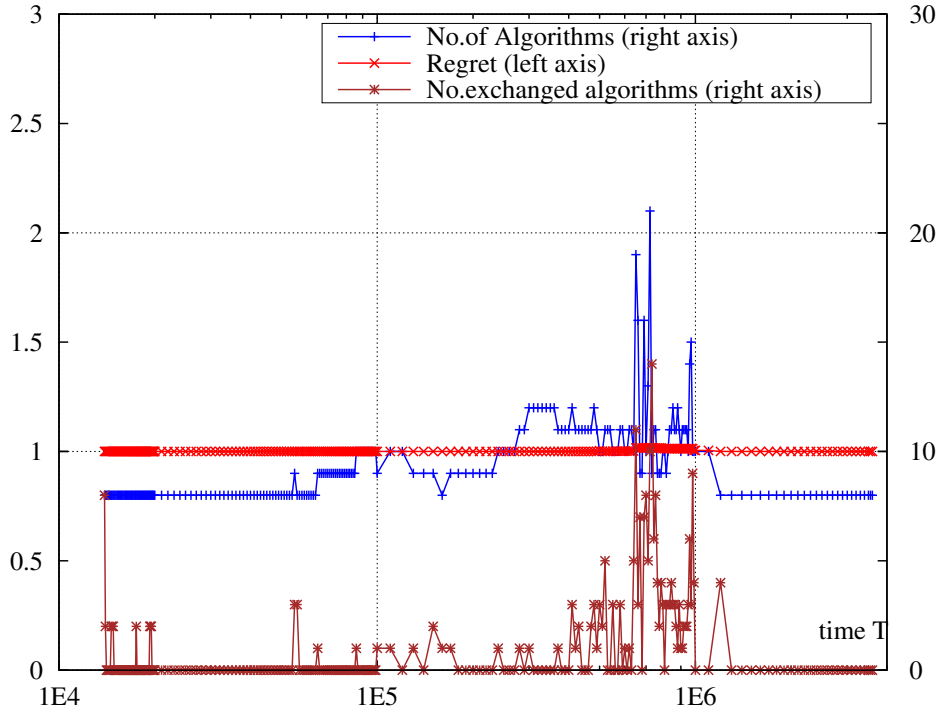
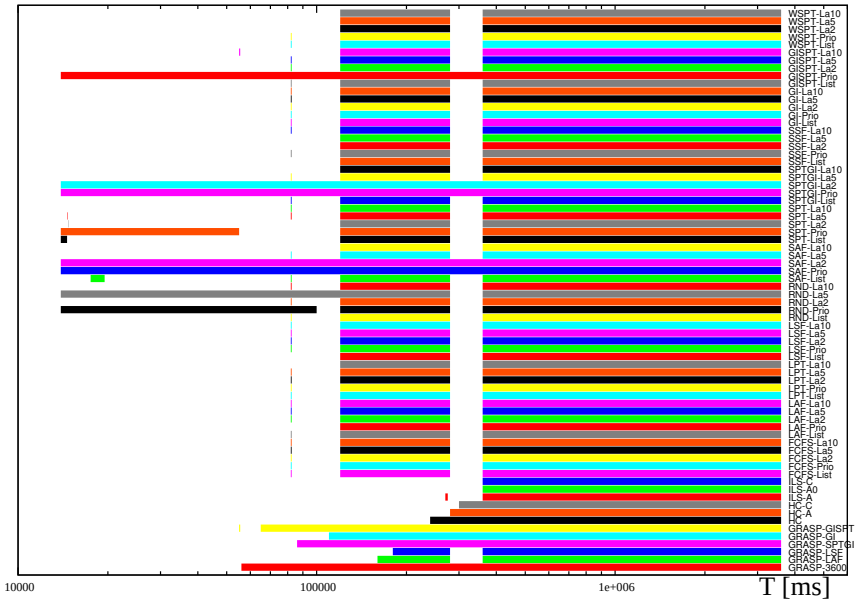


Figure C.14: Algorithm portfolio built on random instances N with $n = 10000$, (dataset 2) $Cost = 8T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

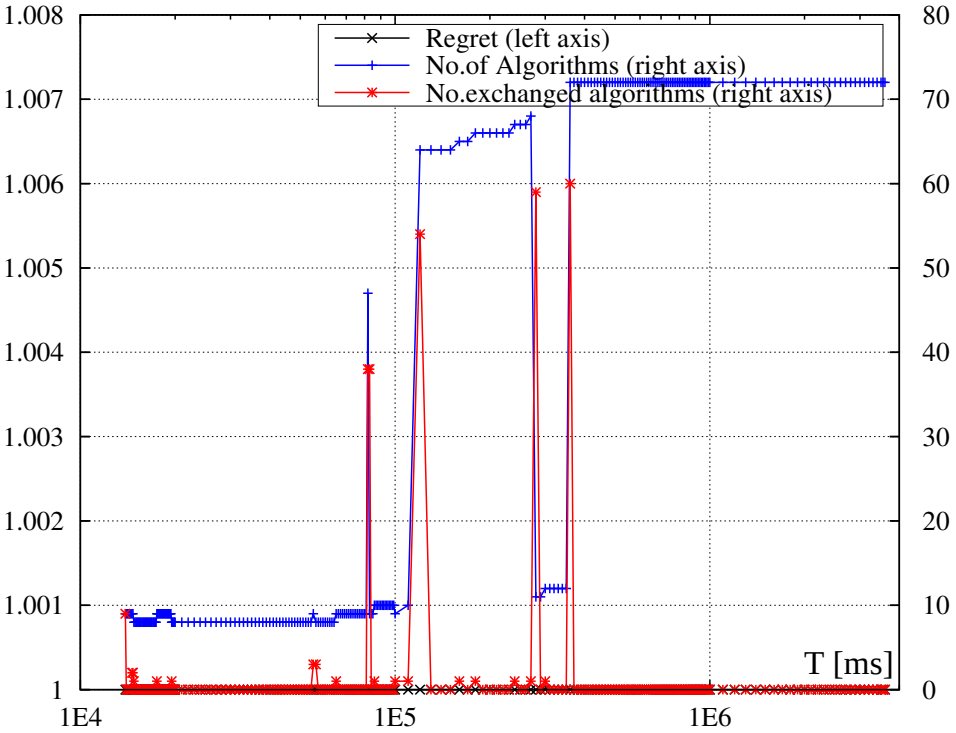
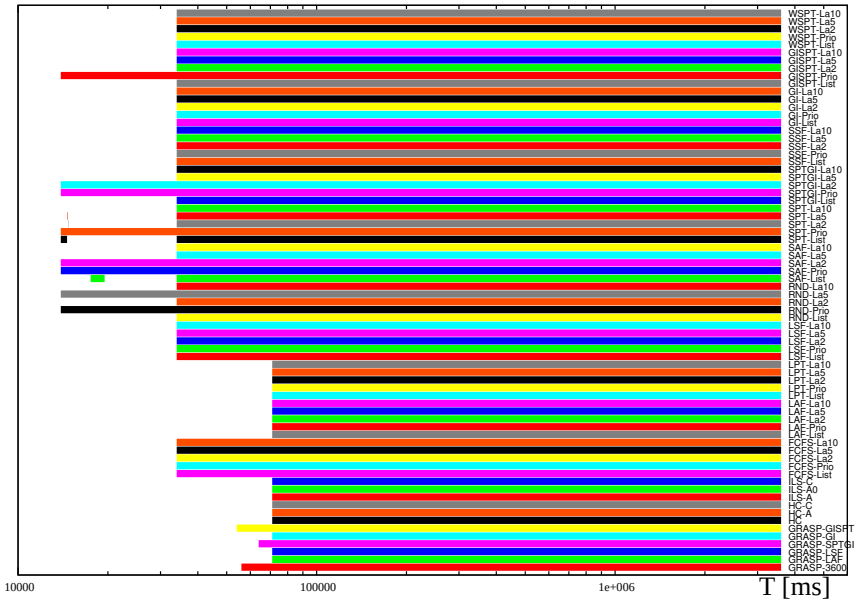


Figure C.15: Algorithm portfolio built on random instances N with $n = 10000$, (dataset 2) $Cost = 16T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

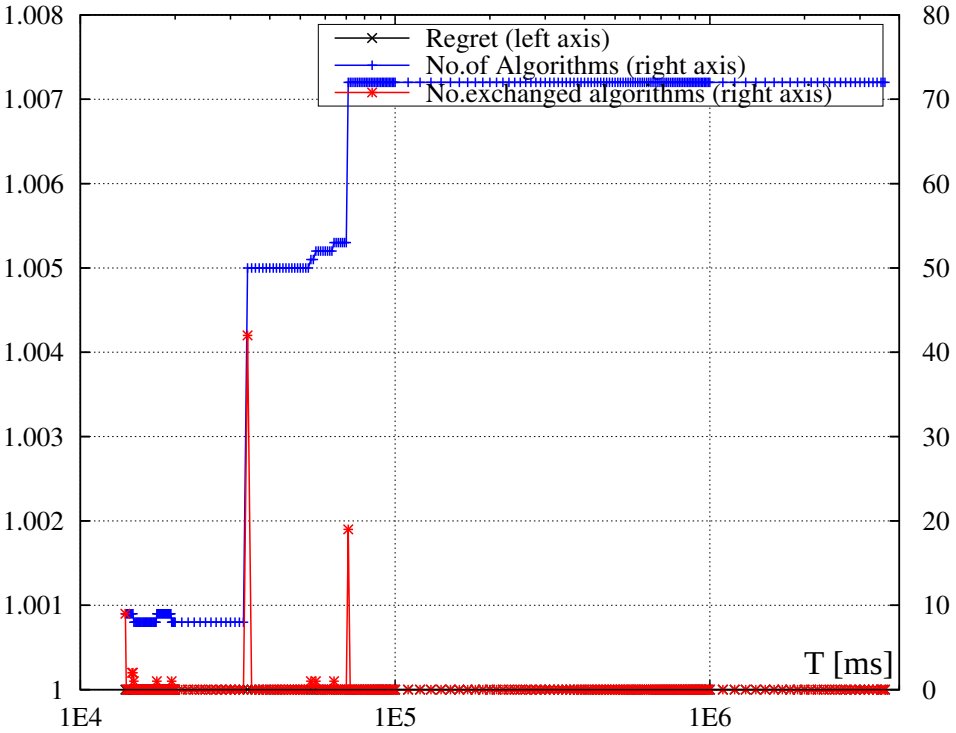
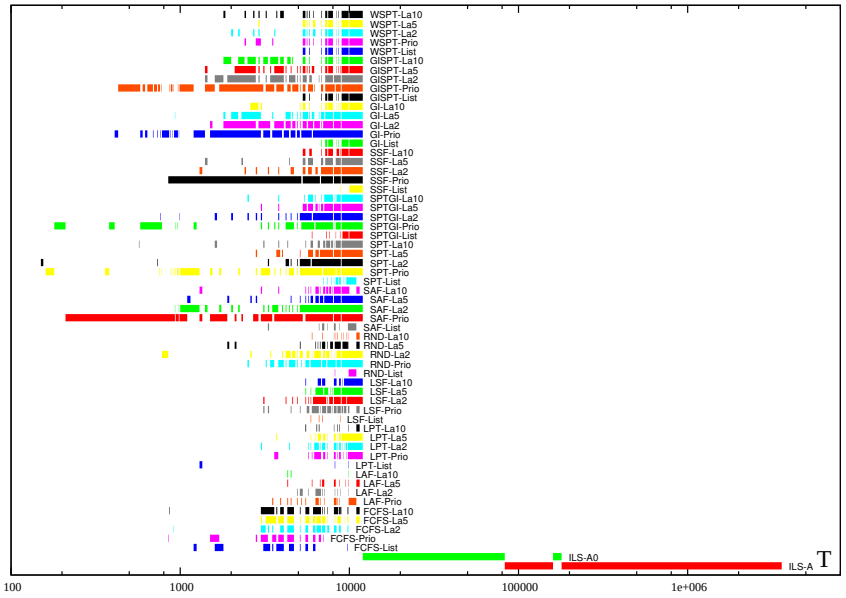


Figure C.16: Algorithm portfolio built on random instances N with $n = 10000$, (dataset 2) $Cost = 32T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

C.5 Regret portfolios for *all* random instances M (dataset 3)

a)



b)

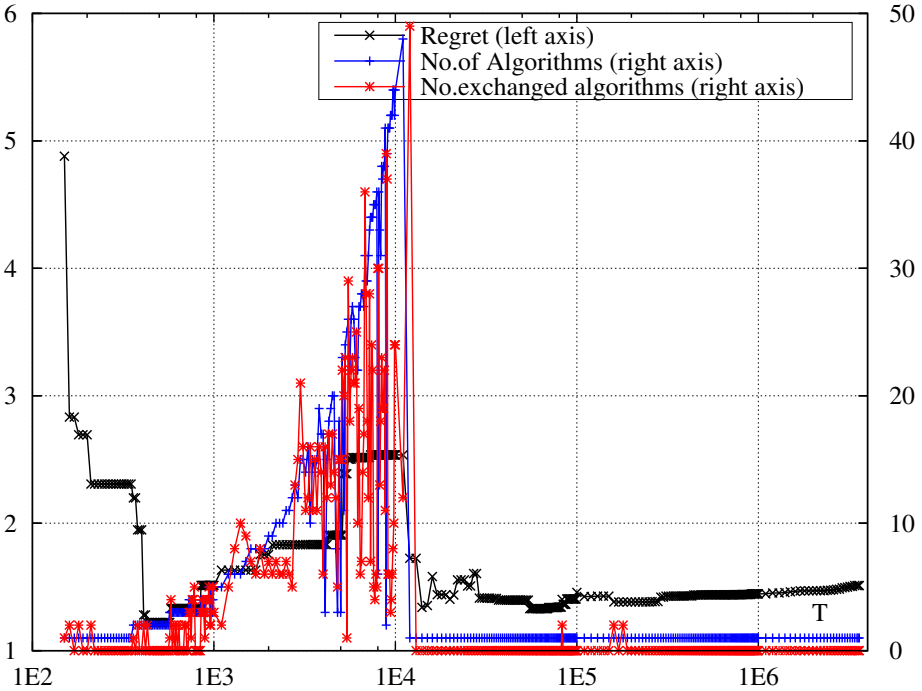
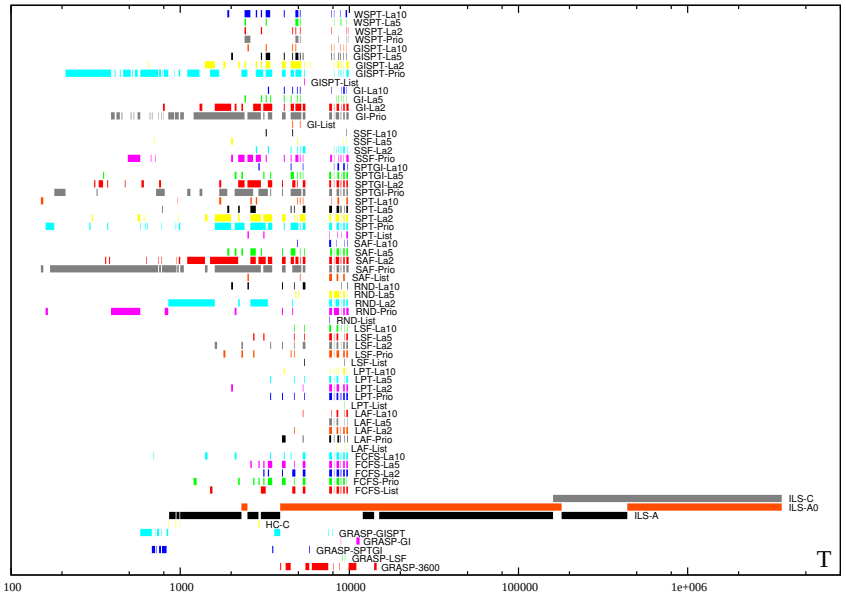


Figure C.17: Algorithm portfolio built on all random instances M (dataset 3), $Cost = 1T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

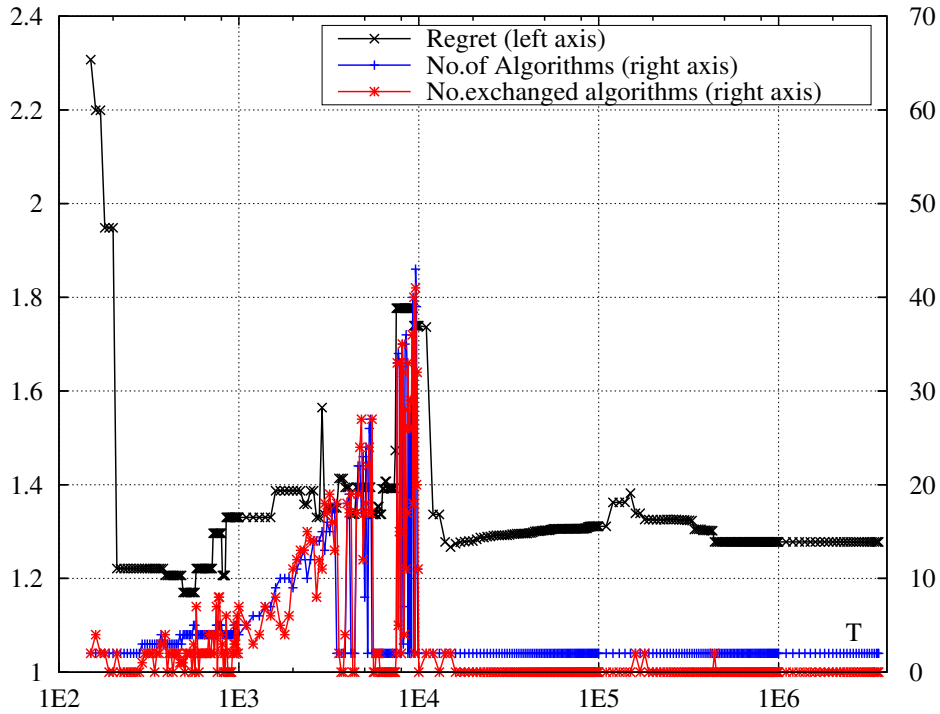
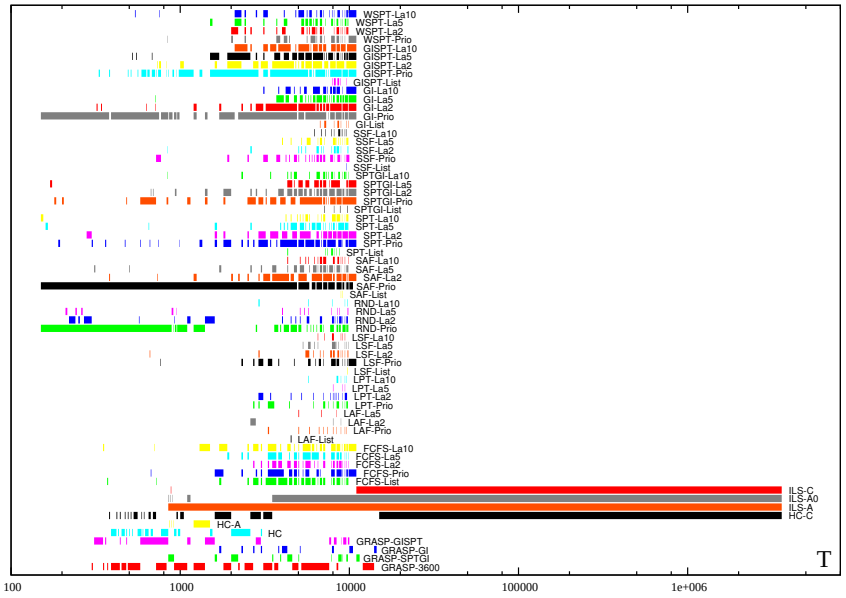


Figure C.18: Algorithm portfolio built on all random instances M (dataset 3), $Cost = 2T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

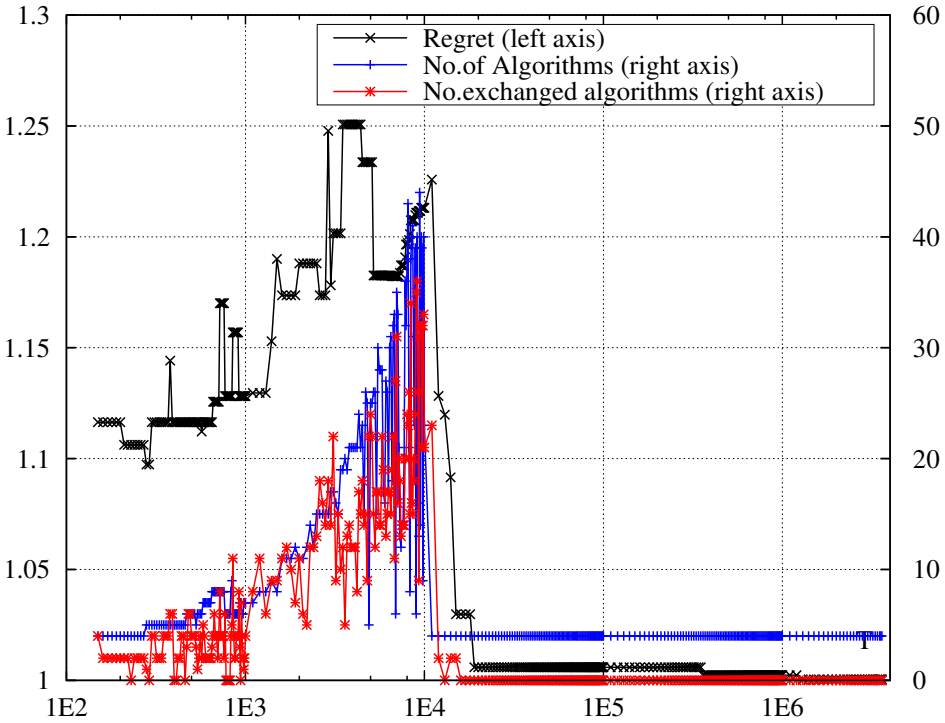
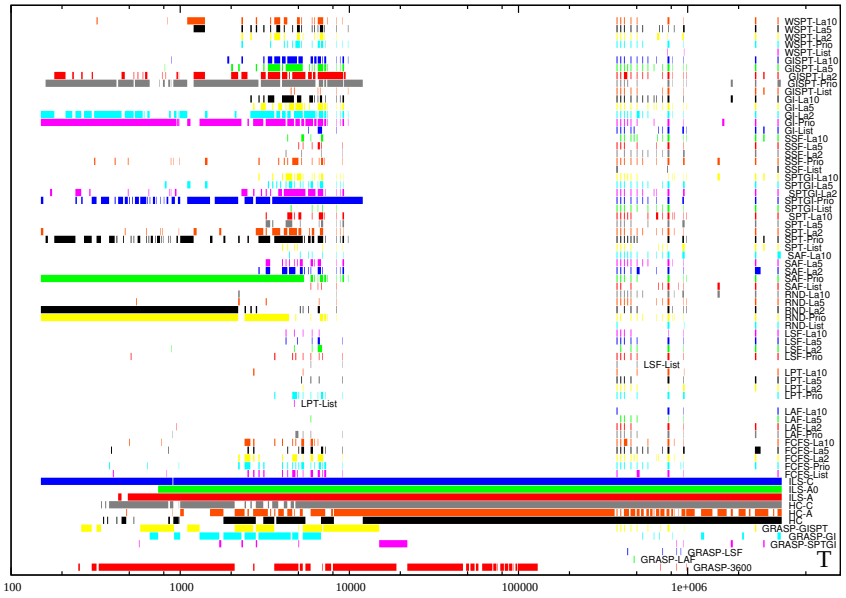


Figure C.19: Algorithm portfolio built on all random instances M (dataset 3), $Cost = 4T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

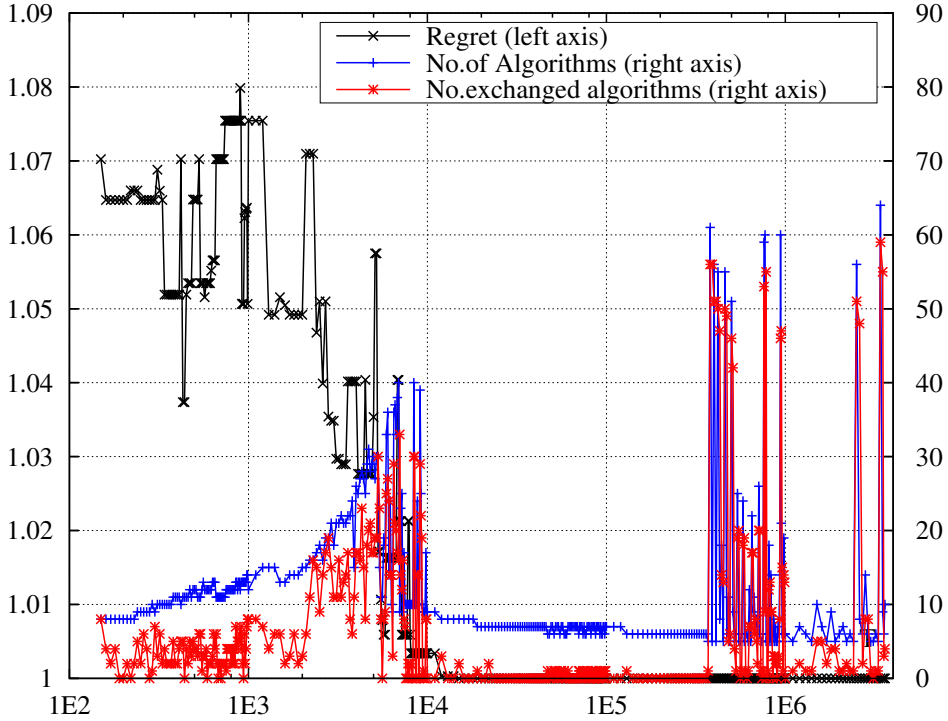
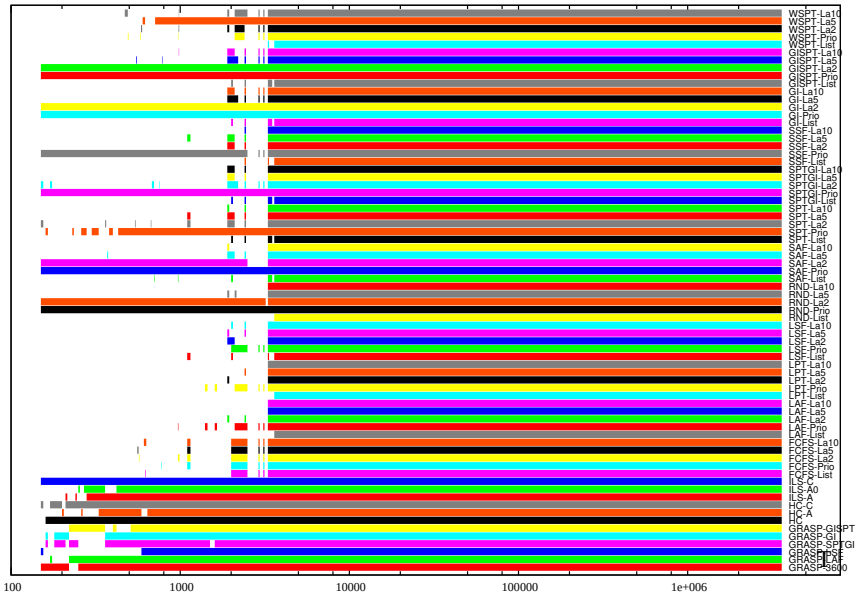


Figure C.20: Algorithm portfolio built on all random instances M (dataset 3), $Cost = 8T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

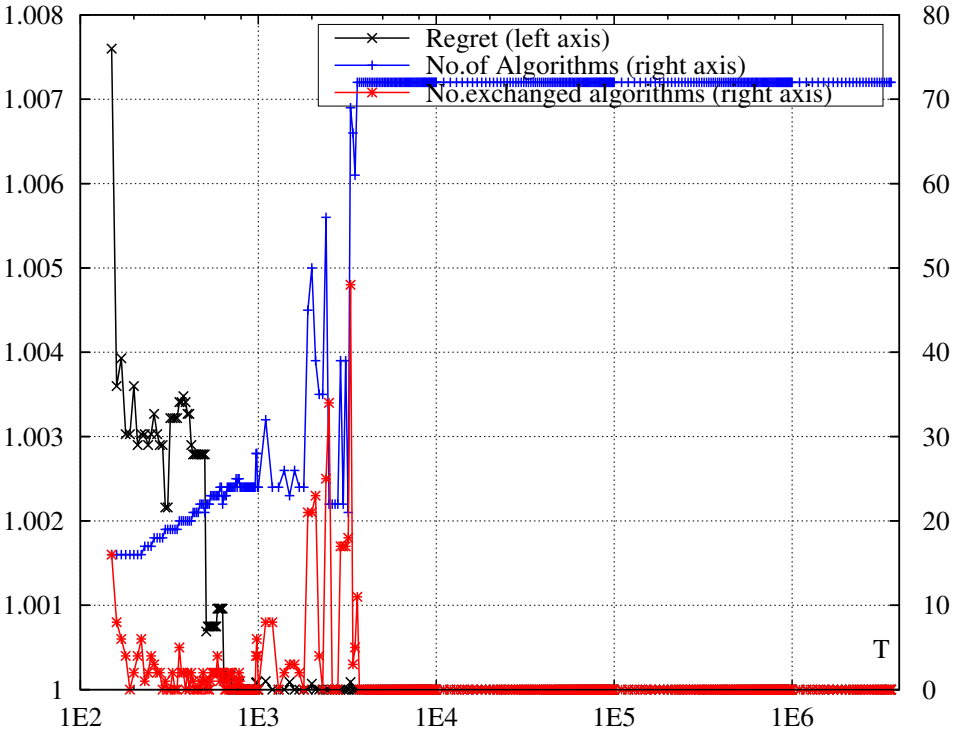
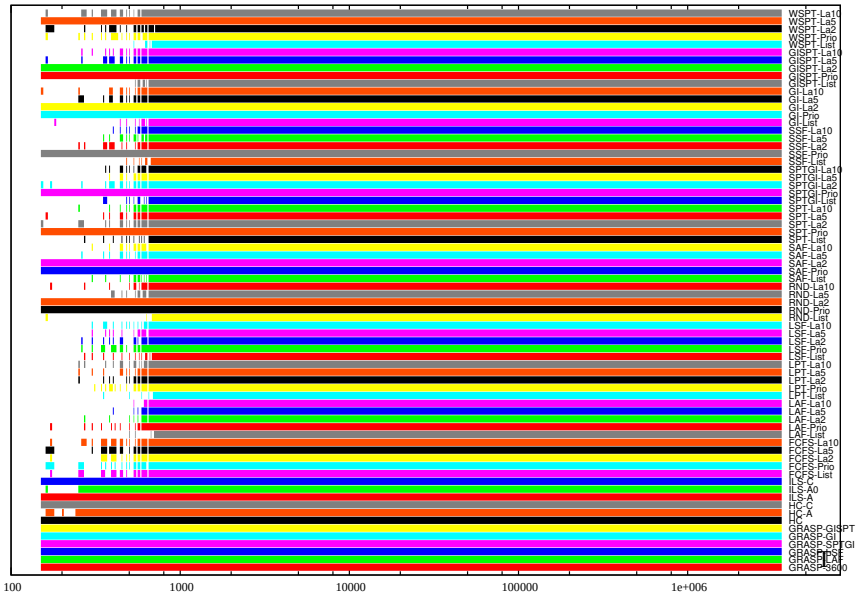


Figure C.21: Algorithm portfolio built on all random instances M (dataset 3), $Cost = 16T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

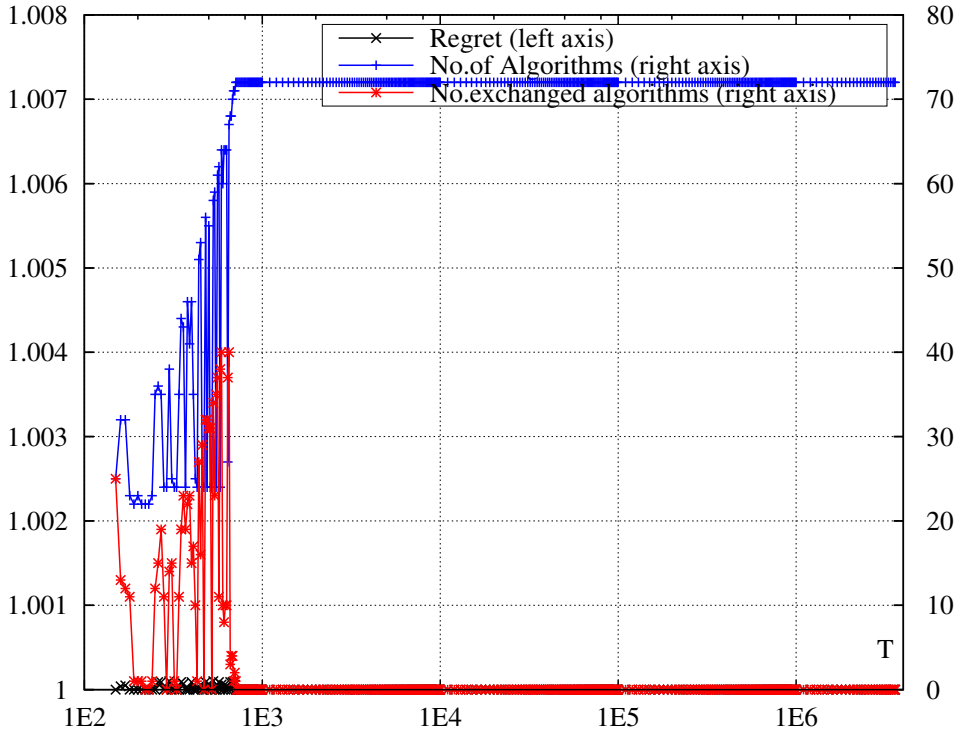
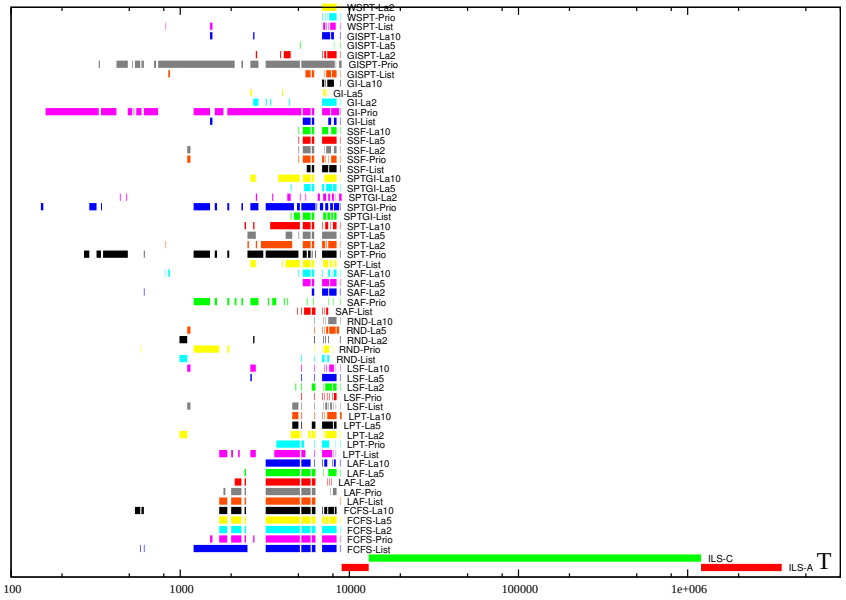


Figure C.22: Algorithm portfolio built on all random instances M (dataset 3), $Cost = 32T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

C.6 Regret portfolios for $m = 2$ random instances M (dataset 4)

a)



b)

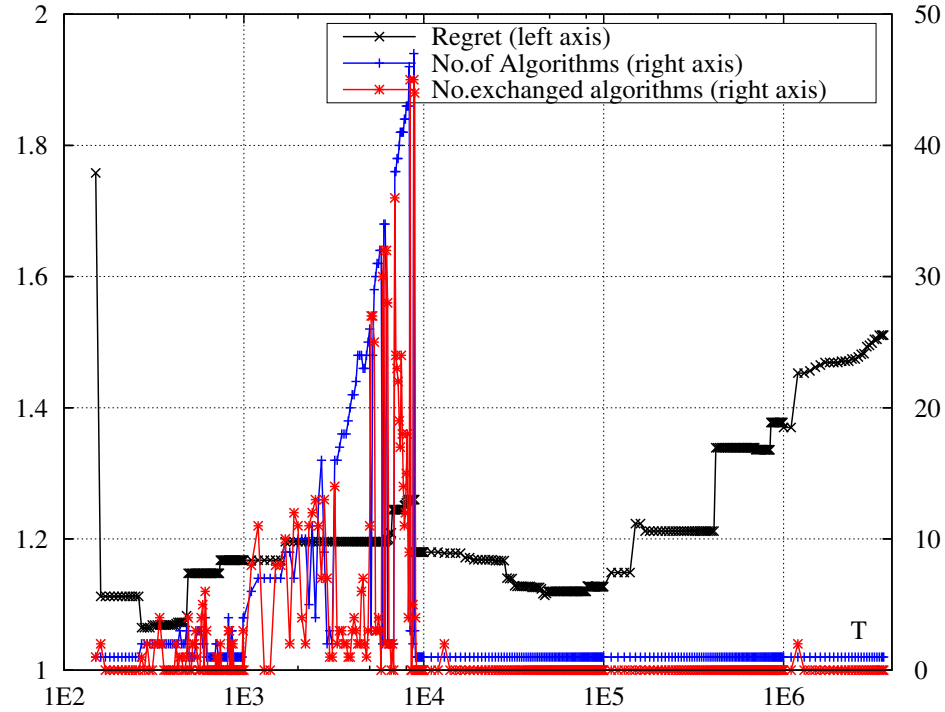
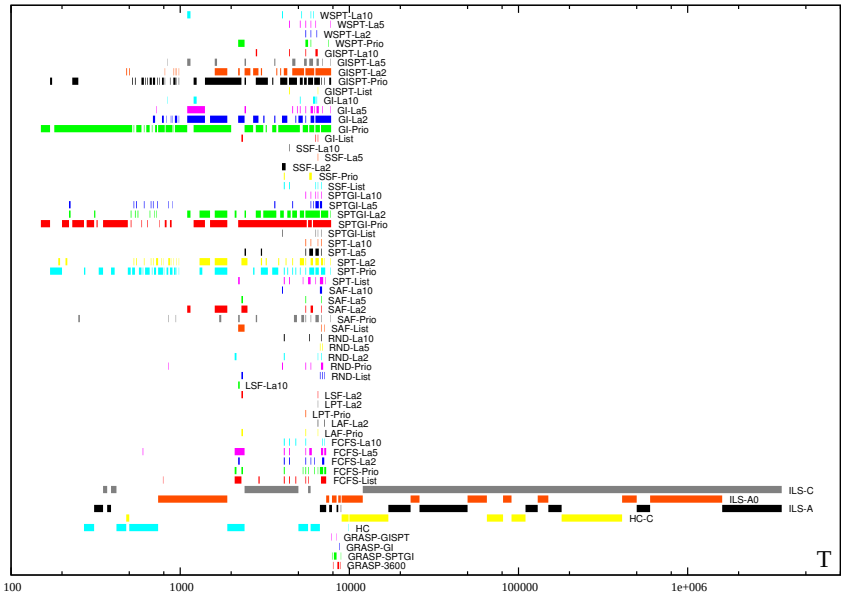


Figure C.23: Algorithm portfolio built on random instances M, with $m = 2$ (dataset 4) $Cost = 1T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

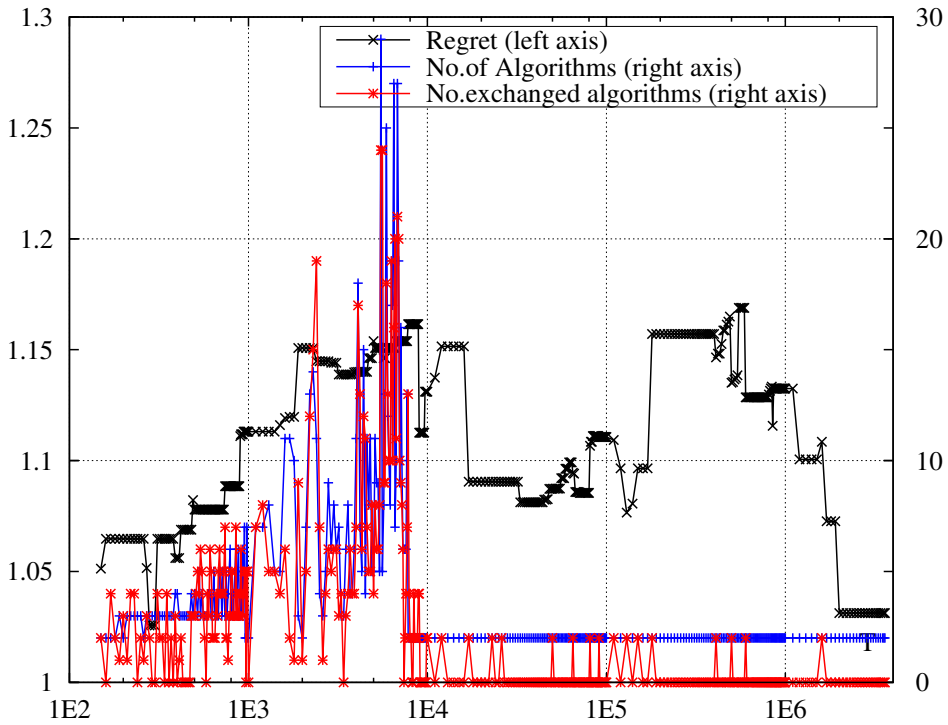
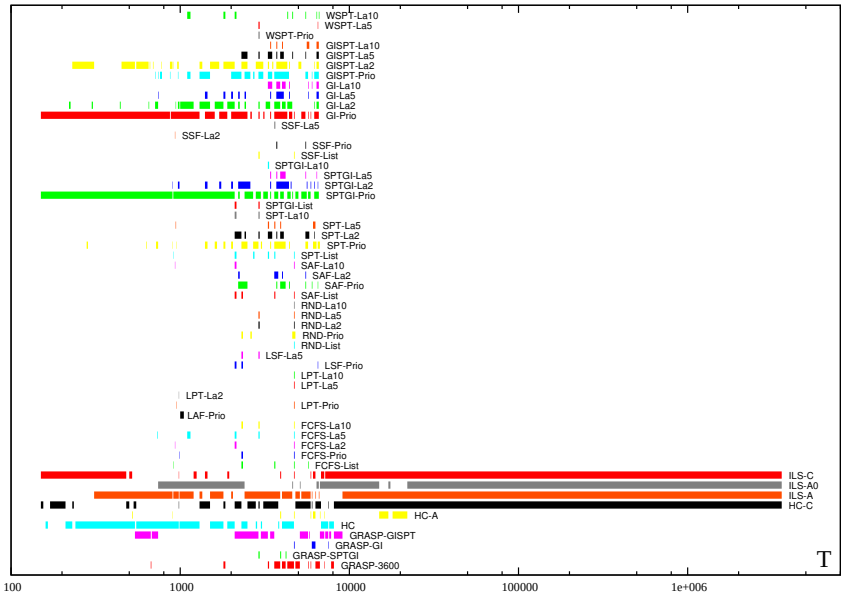


Figure C.24: Algorithm portfolio built on random instances M, with $m = 2$ (dataset 4) $Cost = 2T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

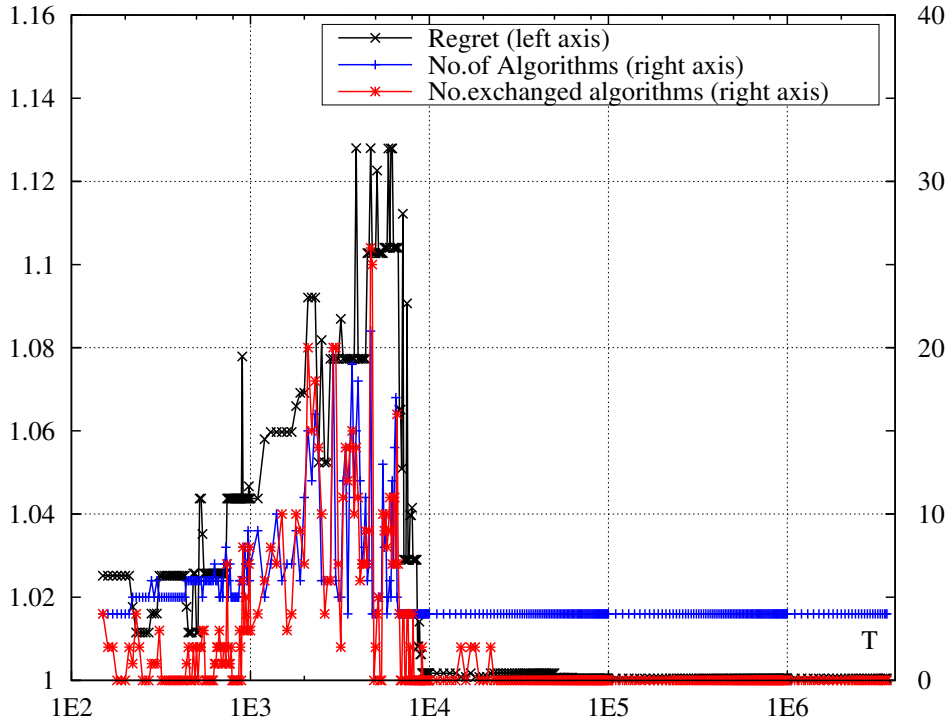
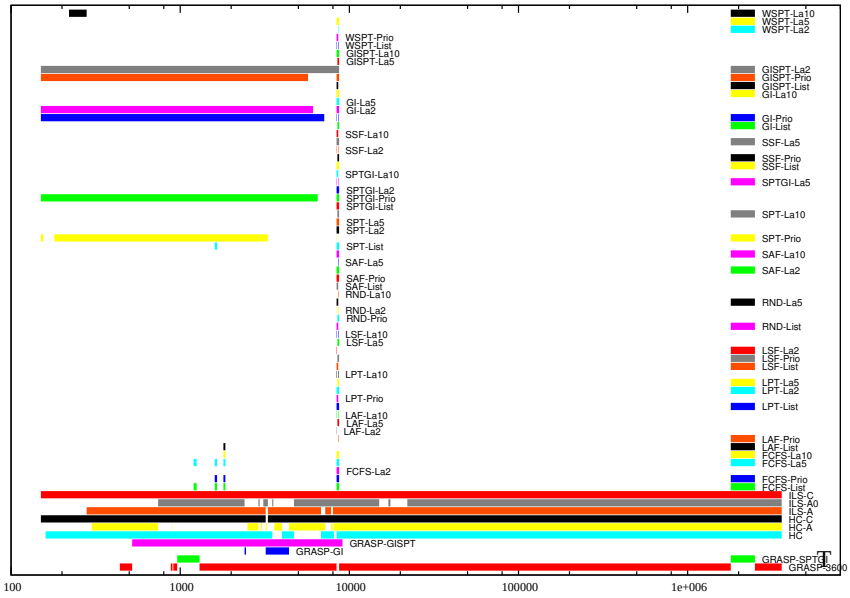


Figure C.25: Algorithm portfolio built on random instances M, with $m = 2$ (dataset 4) $Cost = 4T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

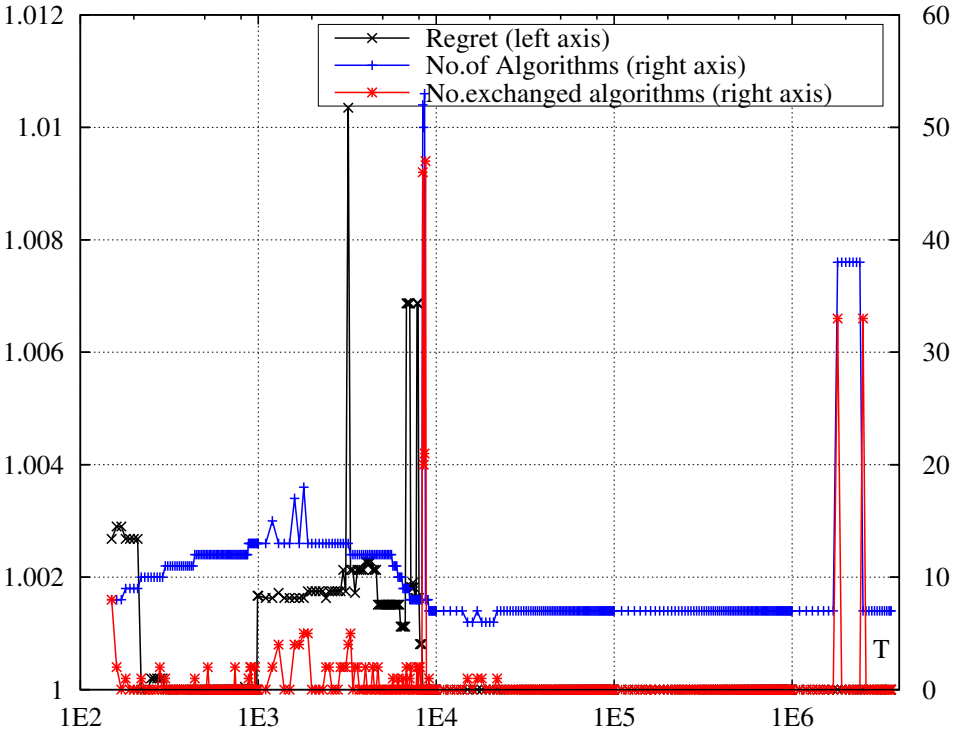
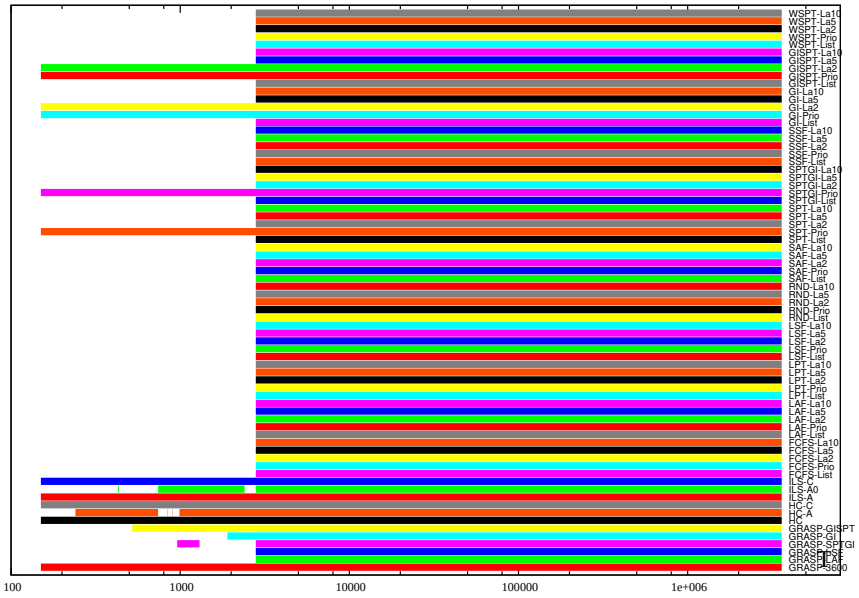


Figure C.26: Algorithm portfolio built on random instances M, with $m = 2$ (dataset 4) $Cost = 8T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

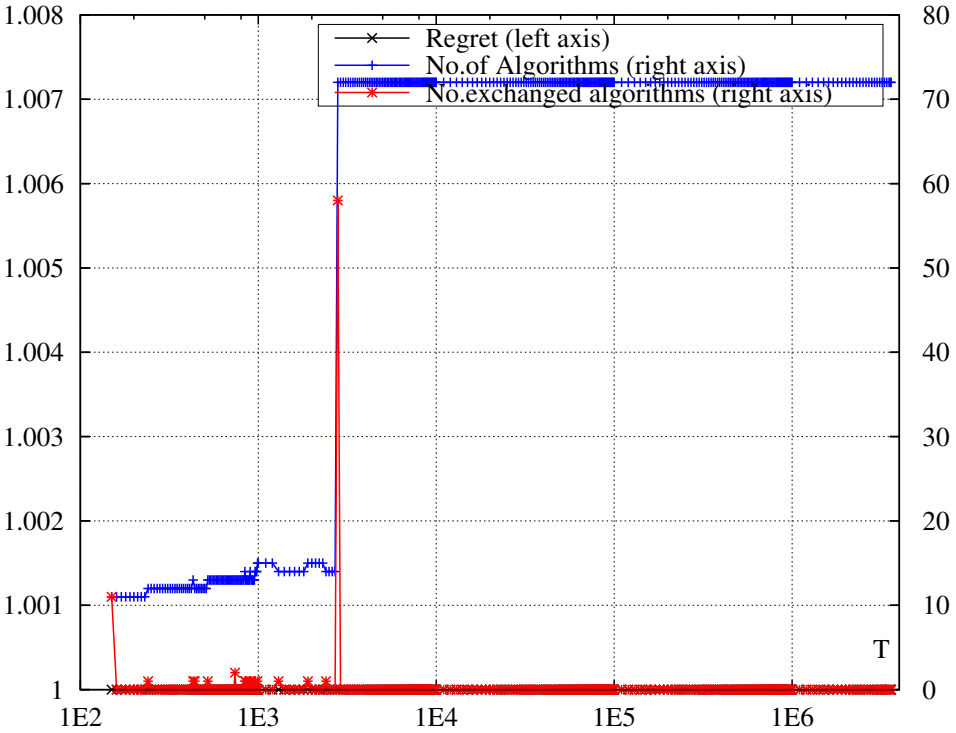
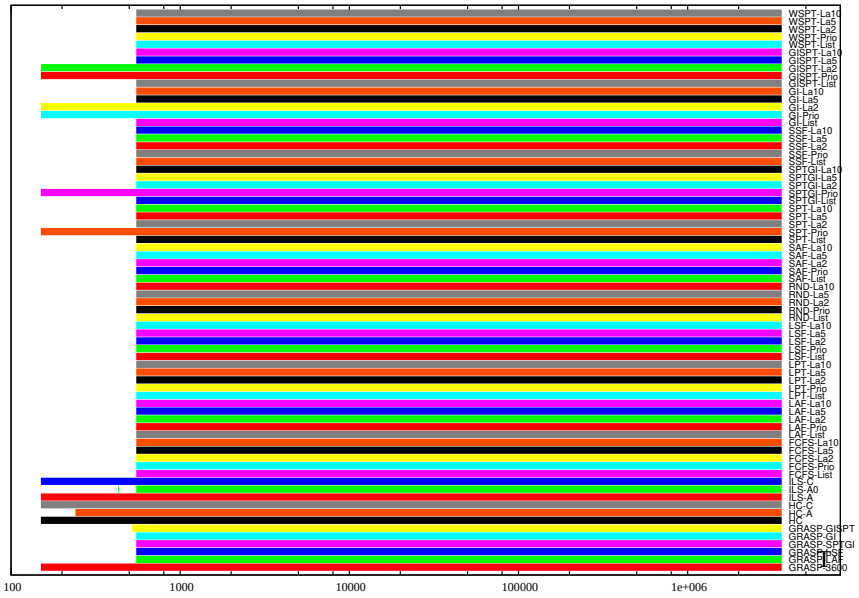


Figure C.27: Algorithm portfolio built on random instances M, with $m = 2$ (dataset 4) $Cost = 16T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

a)



b)

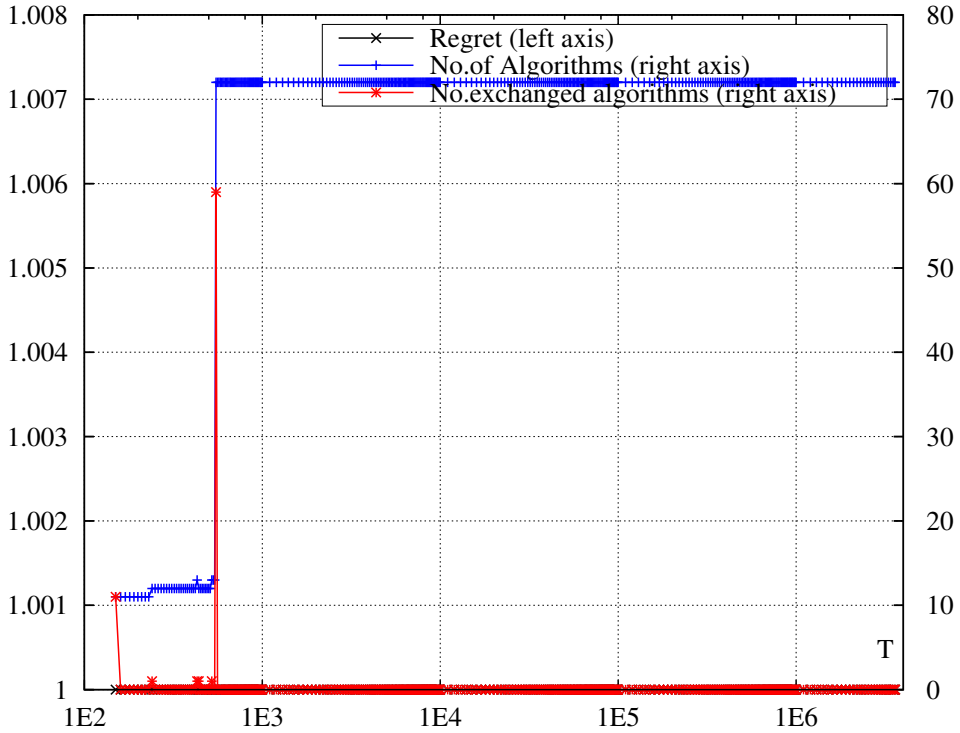
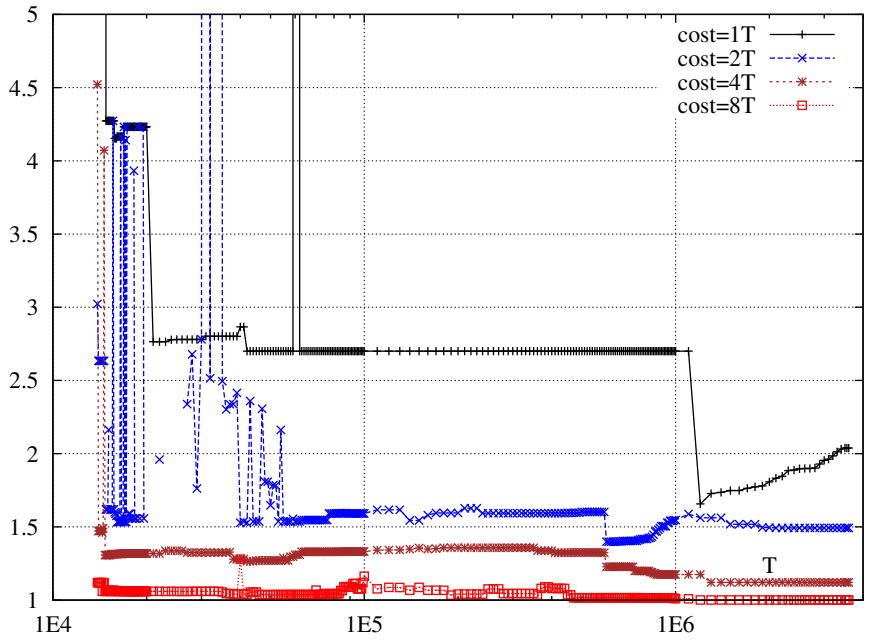


Figure C.28: Algorithm portfolio built on random instances M, with $m = 2$ (dataset 4) $Cost = 32T$. a) portfolio evolution in time T, b) scores of the portfolio (regret, number of algorithms, number of algorithms exchanged), vs T.

C.7 Comparison of regret portfolio scores

In this section regret portfolios scores for $Cost = 1T, 2T, 4T, 8T$ are put together for an easier comparison.

a)



b)

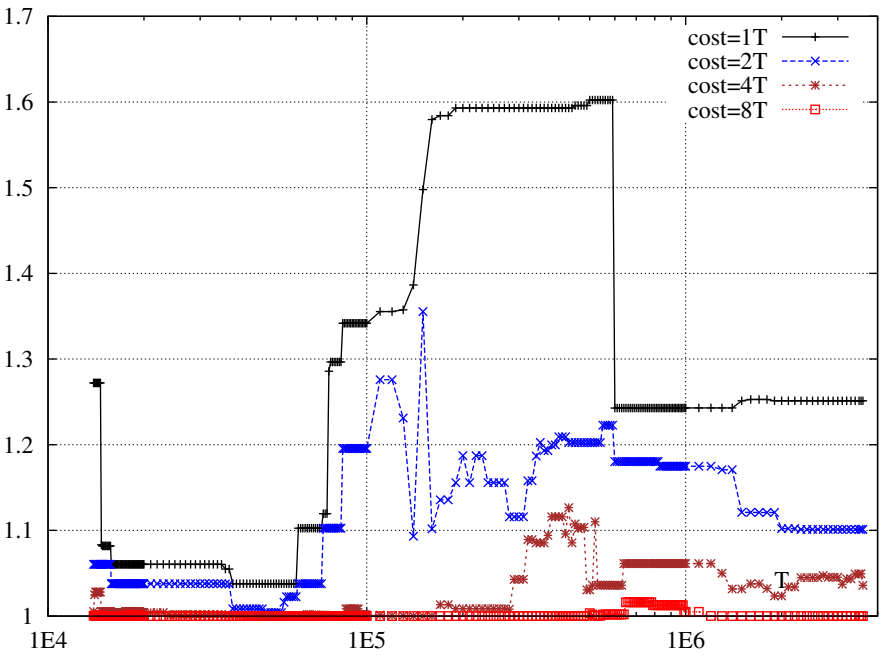
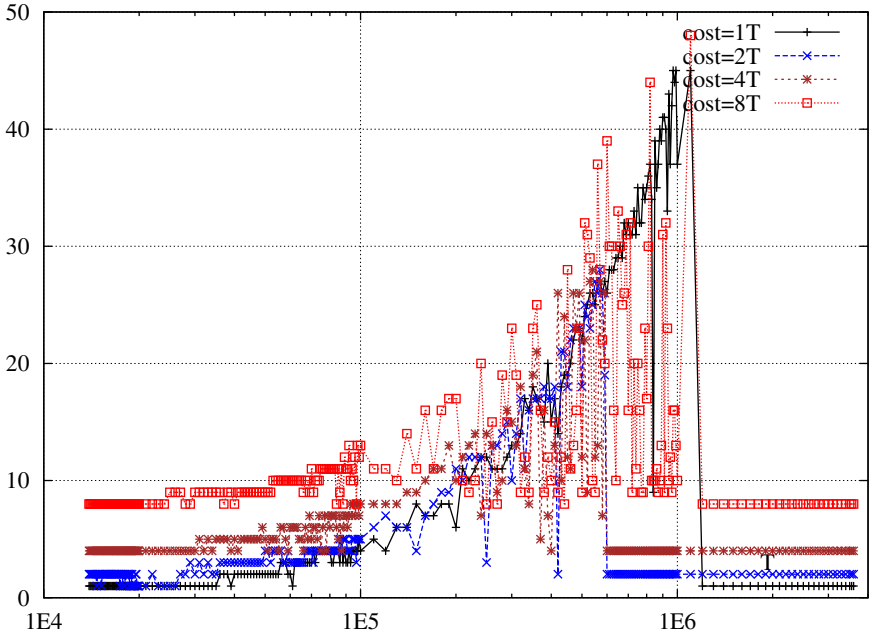


Figure C.29: Comparison of regret portfolio regret scores built on random instances N. a) All random instances N (dataset 1), b) random instances N with $n = 10000$ (dataset 2).

a)



b)

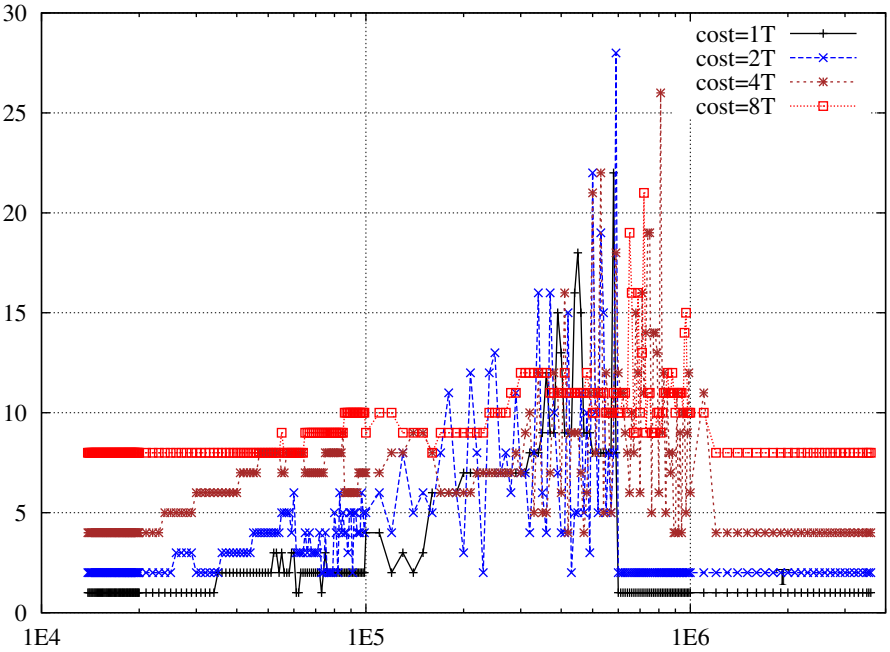
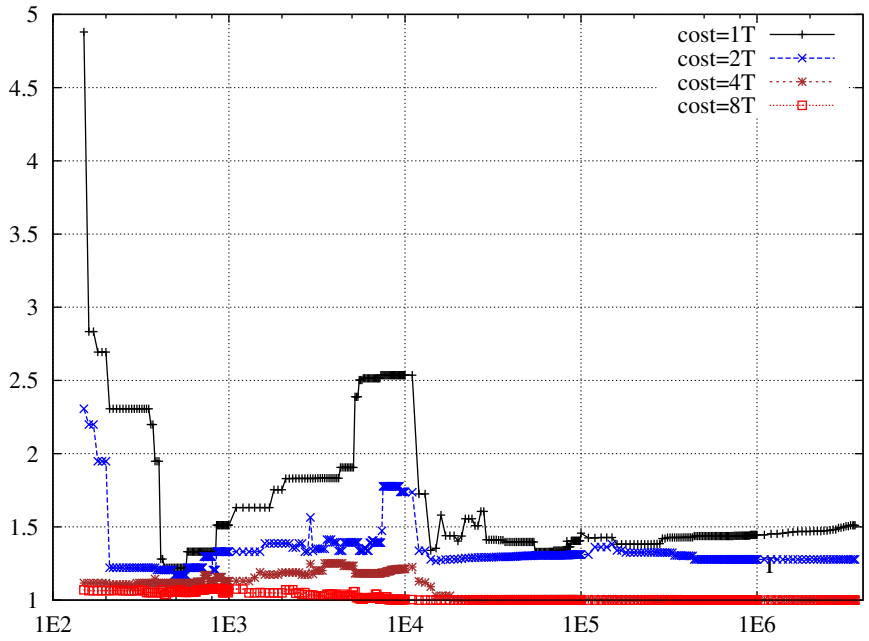


Figure C.30: Comparison of the number of algorithms in the regret portfolios built on random instances N. a) All random instances N (dataset 1), b) random instances N with $n = 10000$ (dataset 2).

a)



b)

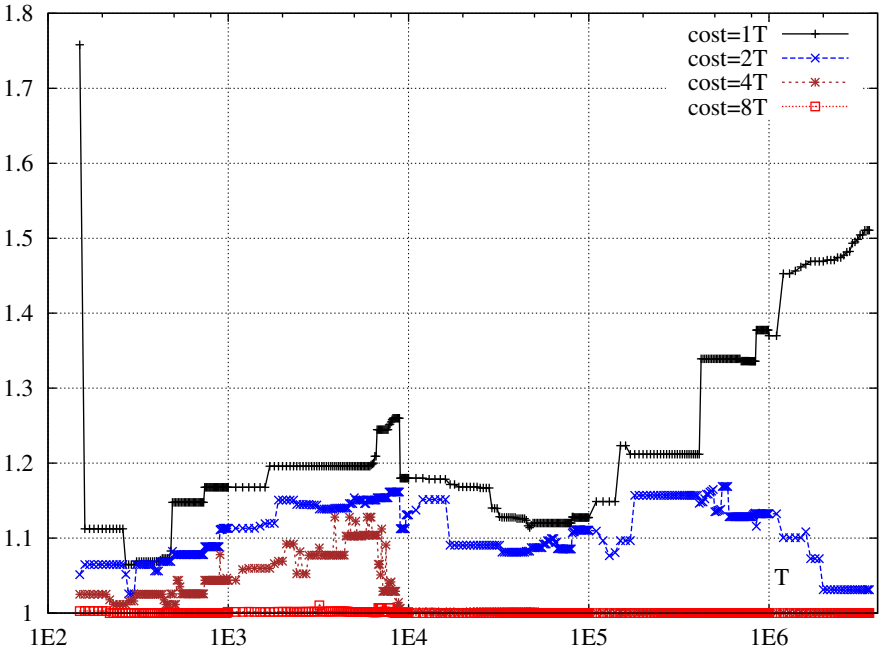
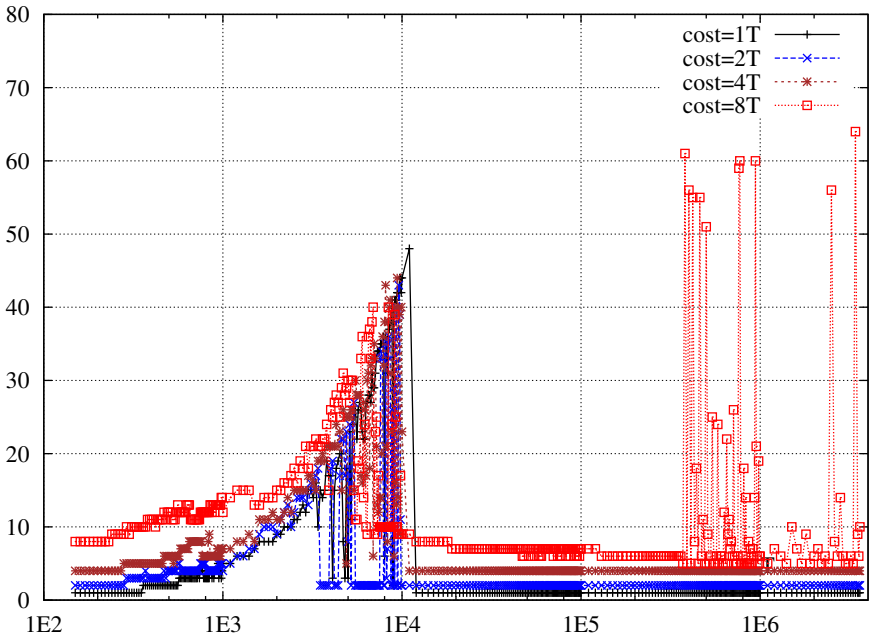


Figure C.31: Comparison of regret portfolio regret scores built on random instances M. a) All random instances M (dataset 3), b) random M with $m = 2$ (dataset 4).

a)



b)

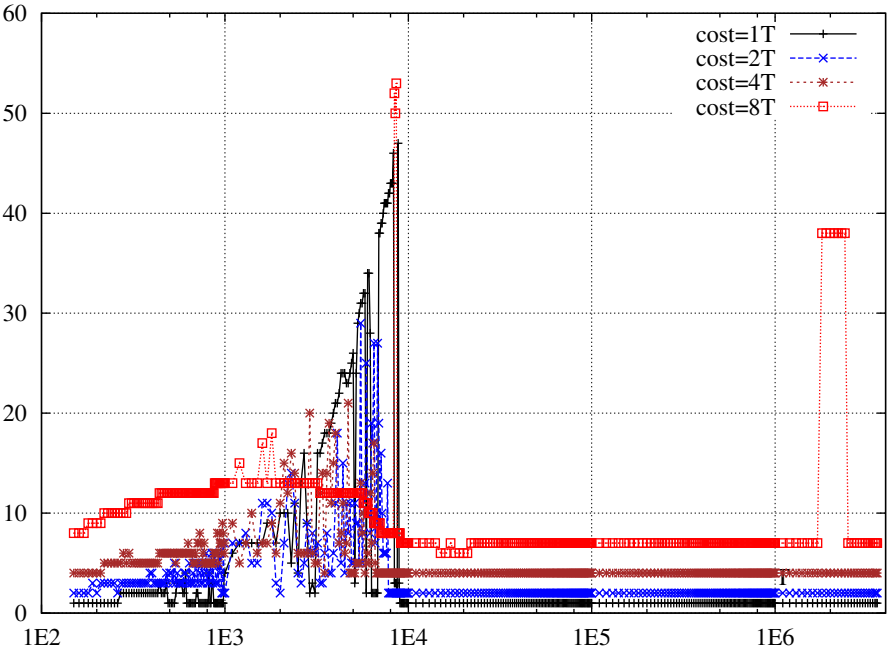


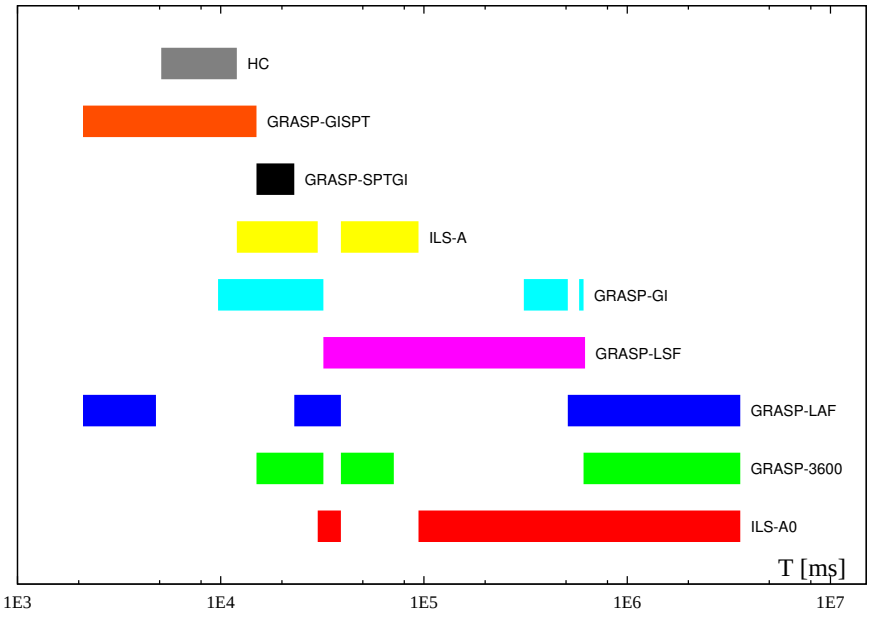
Figure C.32: Comparison of the number of algorithms in the regret portfolios built on random instances M. a) All random instances M (dataset 3), b) random M with $m = 2$ (dataset 4).

C.8 Real Instance Portfolios (dataset 5)

In Fig.C.33b the lines of the cost (in T units) and the number of algorithms overlap. This means that the algorithms in the cover portfolio built for the real instances run in the whole time interval T.

It can be seen if Fig.C.38 and Fig.C.39 the cost limits $16T$ and $32T$ are not really binding for big T, and all algorithms are selected to the portfolio.

a)



b)

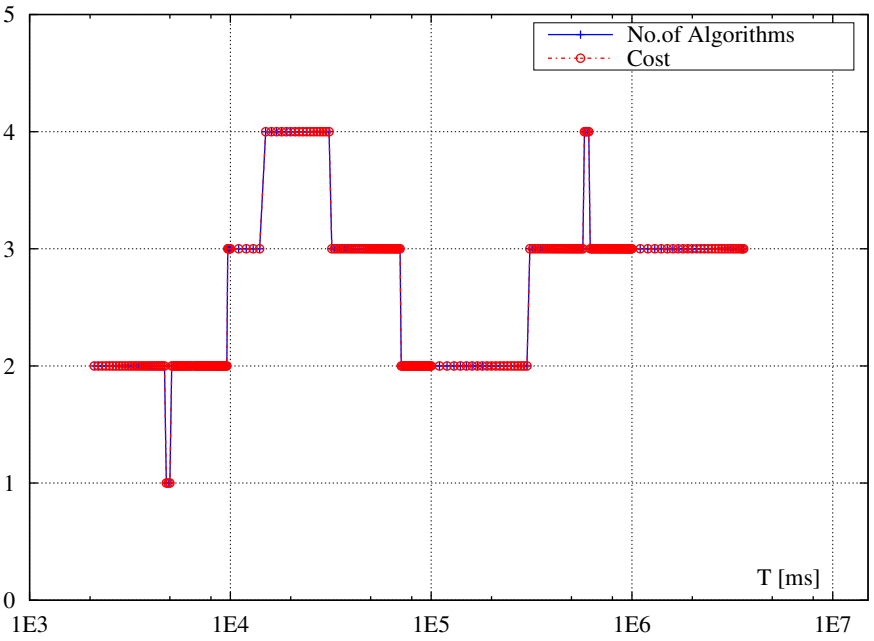
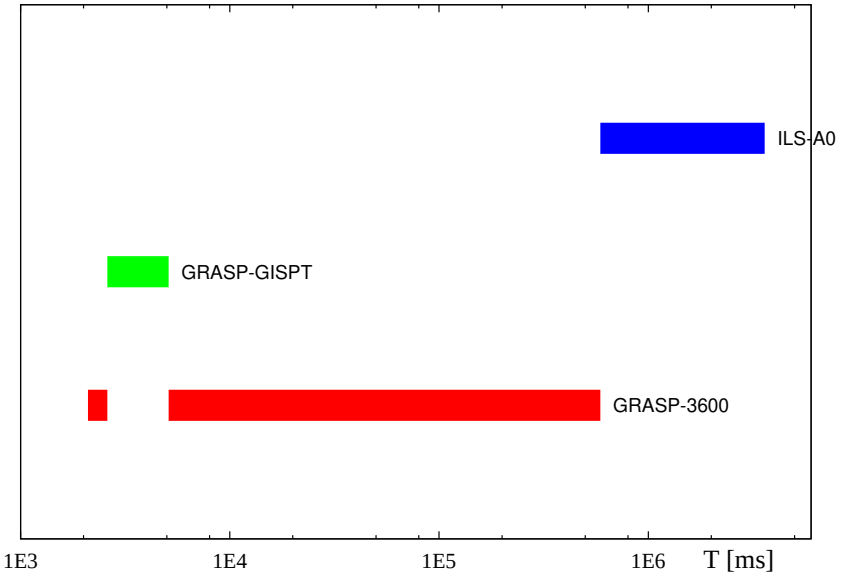


Figure C.33: Cover portfolio built on real instances (dataset 5). a) portfolio evolution in time T, b) scores of the portfolio (number of algorithms, *Cost* in T units, vs T.

a)



b)

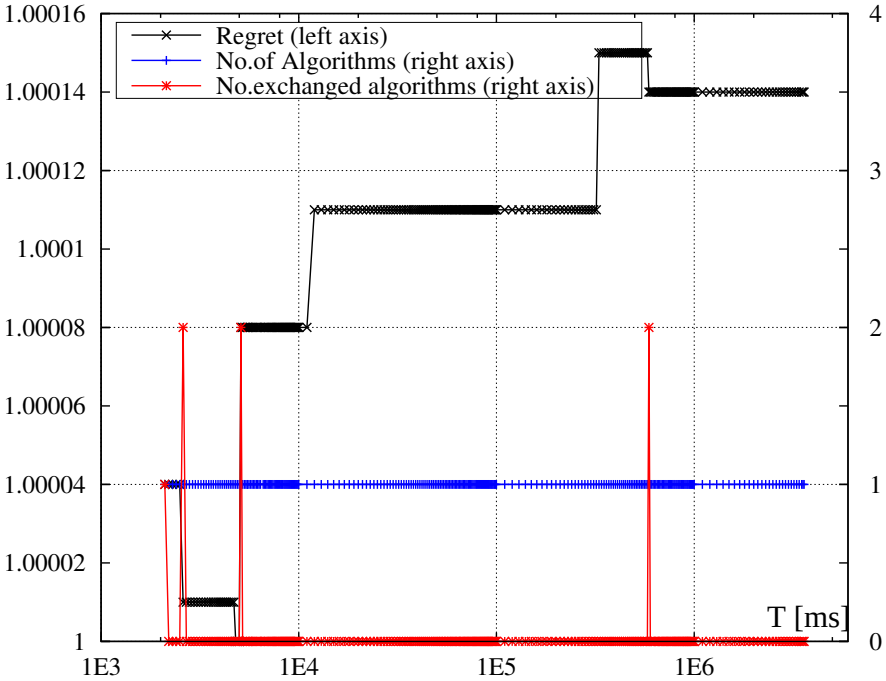
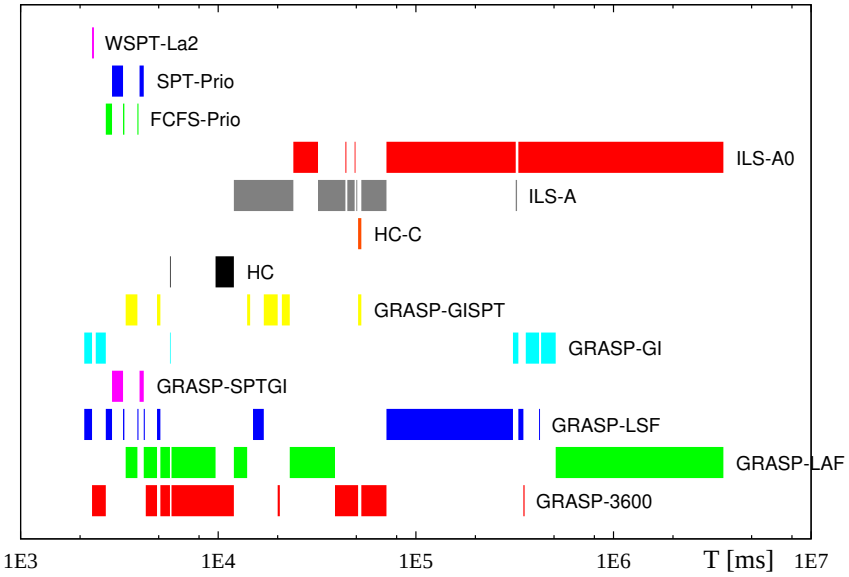


Figure C.34: Regret portfolio built on real instances (dataset 5) $Cost = 1T$. a) portfolio evolution in time T, b) scores of the portfolio (number of algorithms, $Cost$ in T units, number of algorithms exchanged algorithms) vs T.

a)



b)

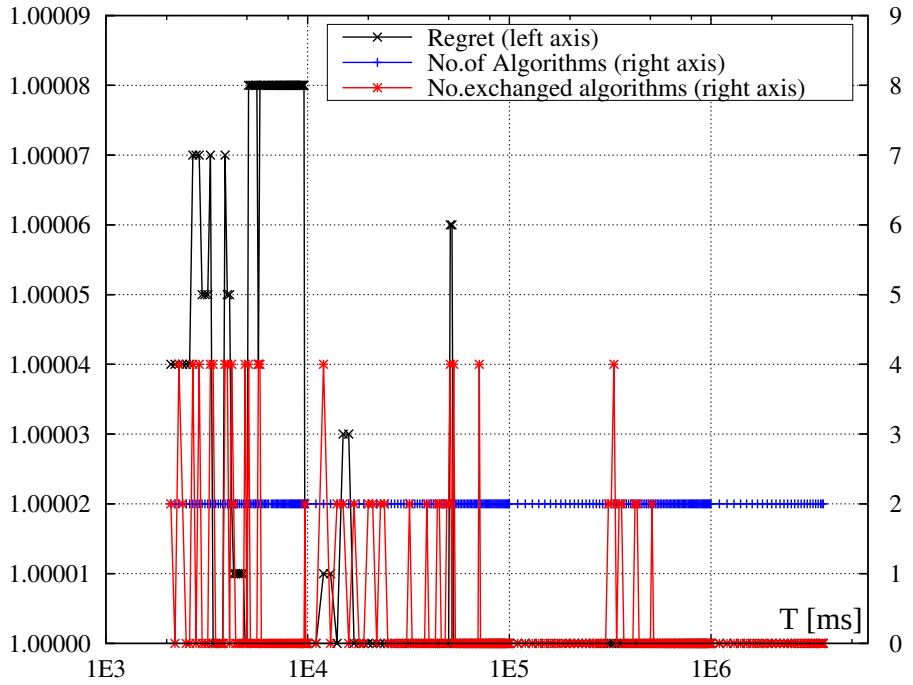
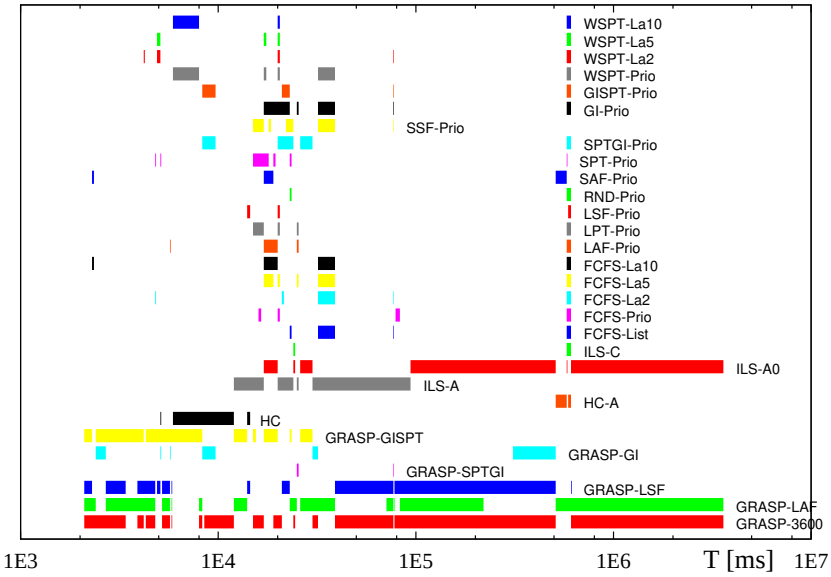


Figure C.35: Regret portfolio built on real instances (dataset 5) $Cost = 2T$. a) portfolio evolution in time T, b) scores of the portfolio (number of algorithms, $Cost$ in T units, number of algorithms exchanged algorithms) vs T.

a)



b)

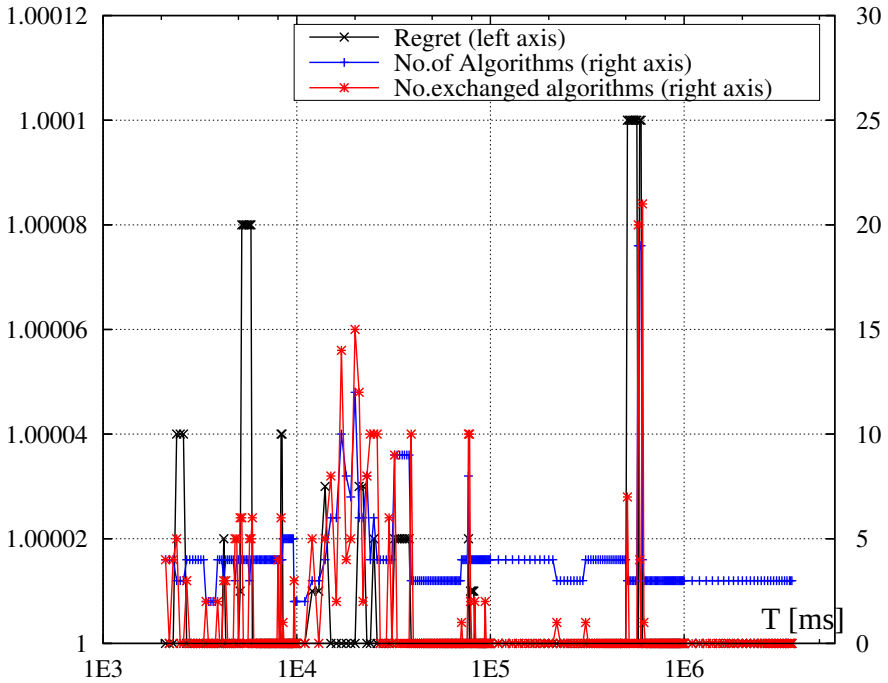
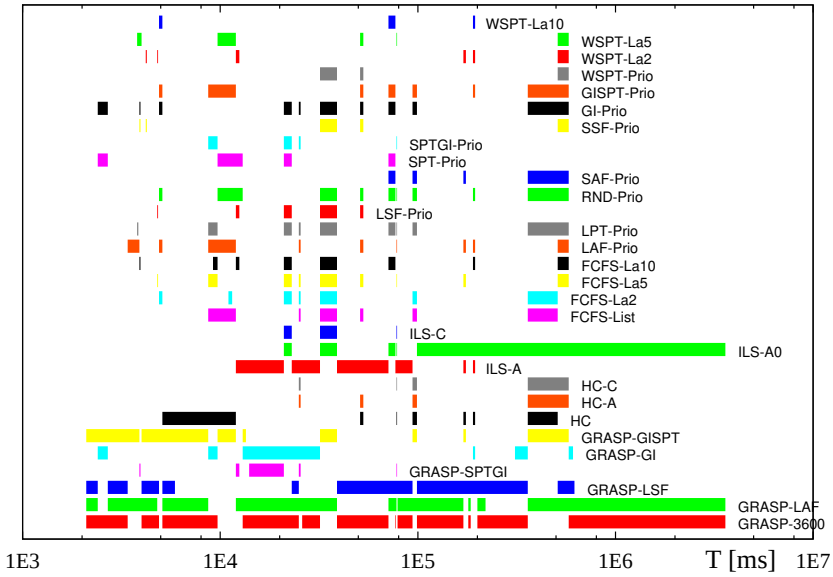


Figure C.36: Regret portfolio built on real instances (dataset 5) $Cost = 4T$. a) portfolio evolution in time T, b) scores of the portfolio (number of algorithms, $Cost$ in T units, number of algorithms exchanged algorithms) vs T.

a)



b)

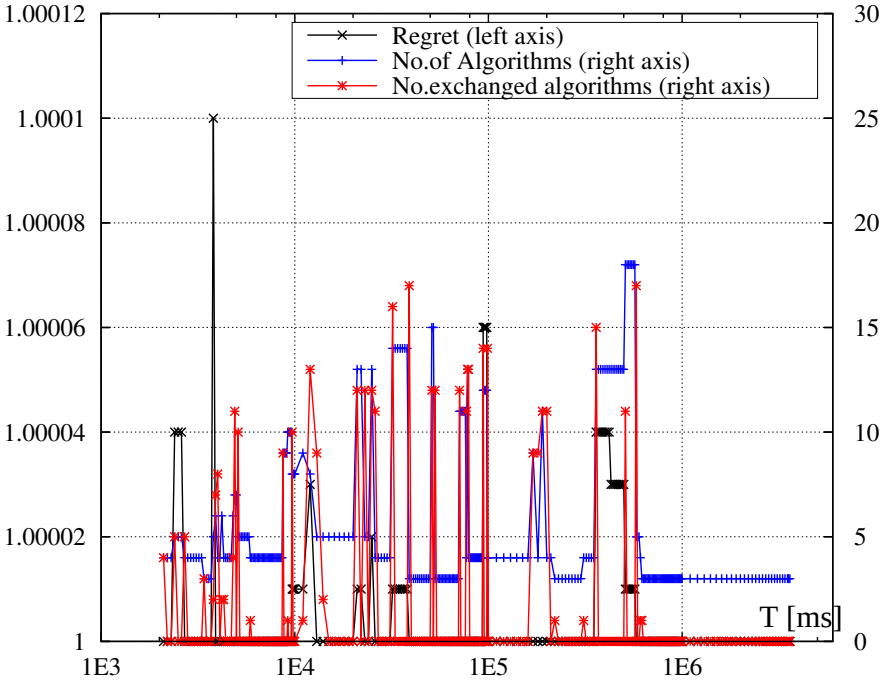
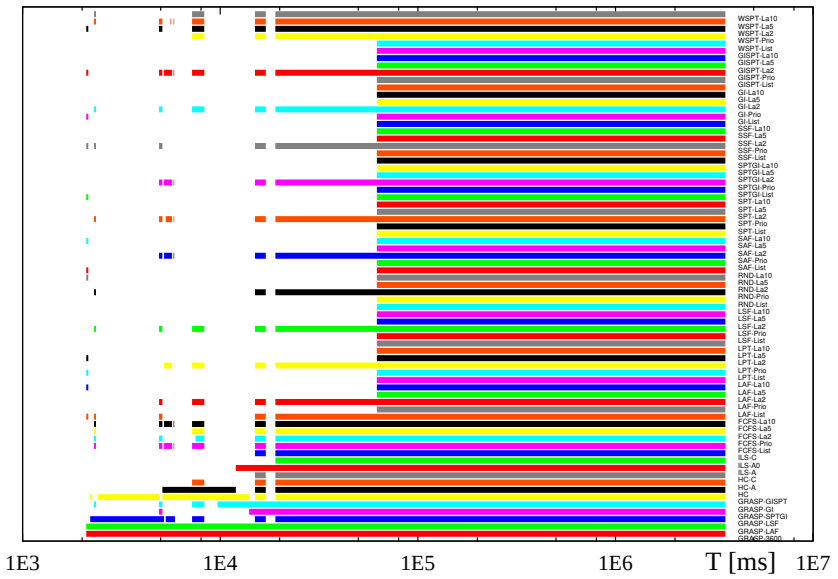


Figure C.37: Regret portfolio built on real instances (dataset 5) $Cost = 8T$. a) portfolio evolution in time T, b) scores of the portfolio (number of algorithms, $Cost$ in T units, number of algorithms exchanged algorithms) vs T.

a)



b)

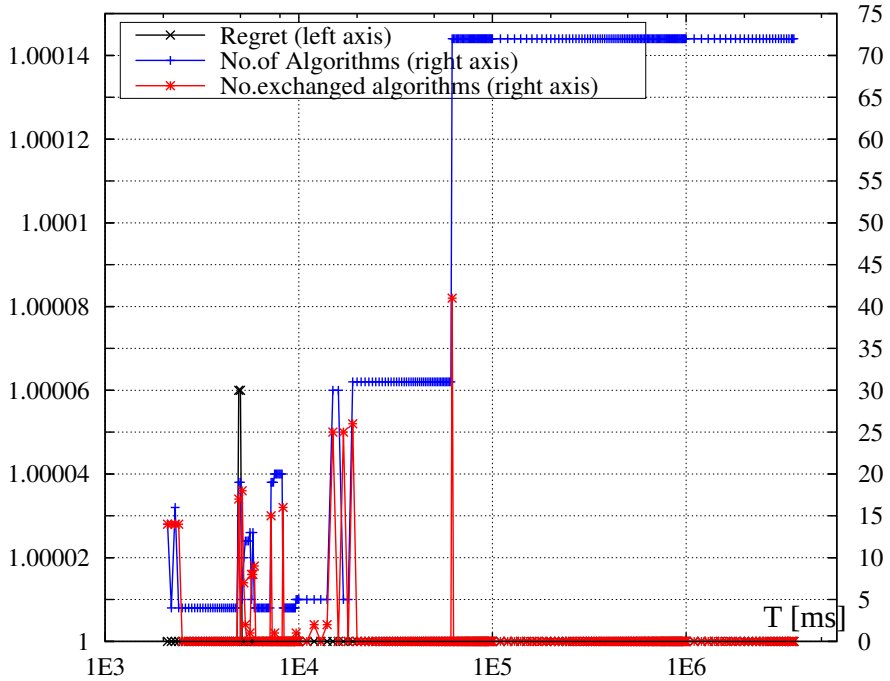
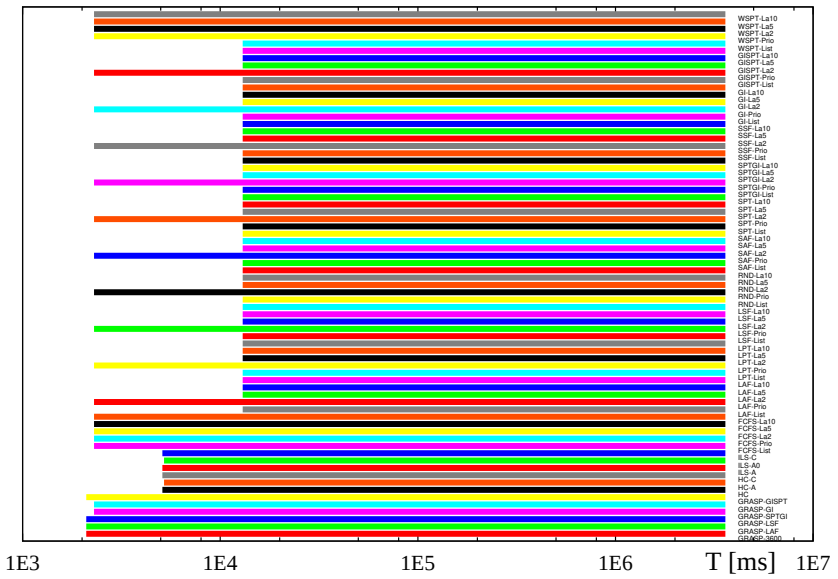


Figure C.38: Regret portfolio built on real instances (dataset 5) $Cost = 16T$. a) portfolio evolution in time T, b) scores of the portfolio (number of algorithms, $Cost$ in T units, number of algorithms exchanged algorithms) vs T.

a)



b)

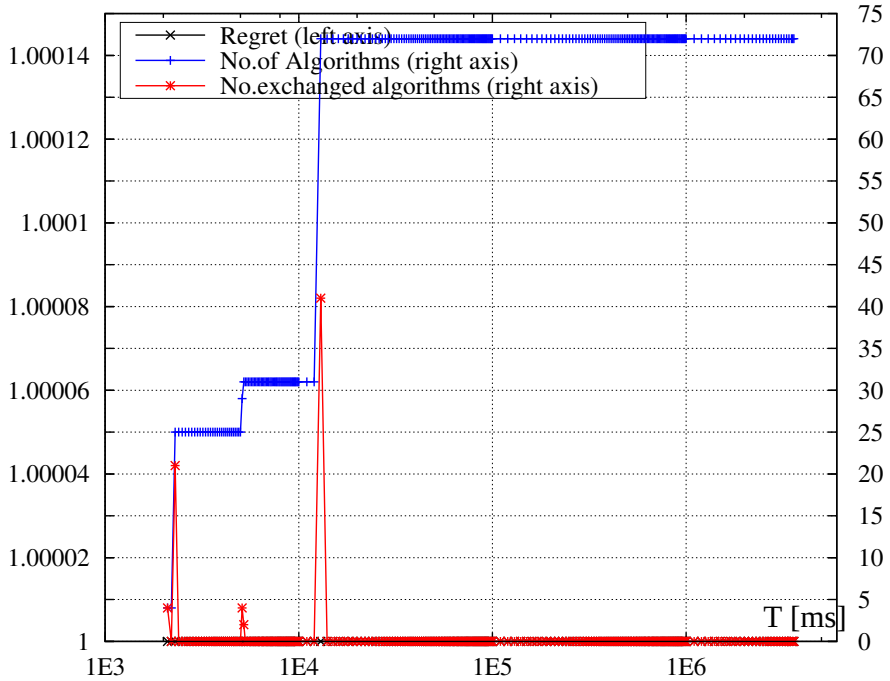


Figure C.39: Regret portfolio built on real instances (dataset 5) $Cost = 32T$. a) portfolio evolution in time T, b) scores of the portfolio (number of algorithms, $Cost$ in T units, number of algorithms exchanged algorithms) vs T.

Appendix D

Ship Traffic Model – Distributions and Parameters

D.1 Basic dataset information

The model was developed on the basis of data for maritime container terminals in Gdańsk, Long Beach, Los Angeles, Le Havre, Hamburg, Rotterdam, Shanghai, Singapore in year 2016. Basic dataset information is collected in Tab. D.1. L_j are ship lengths. Values η_a denote the fraction of aperiodic (non-returning) arrivals.

Table D.1: Basic dataset information

port	Gdańsk	Long Beach	Los Angeles	Le Havre
number of calls N	471	995	1308	2271
physical ships n	81	242	309	559
returning ships	57	188	238	417
No.of unique L_js	31	65	61	105
η_a[%]	5.10	5.43	5.43	6.25
port	Hamburg	Rotterdam	Shanghai	Singapore
number of calls N	3294	3998	11606	18494
physical ships n	586	311	1233	1857
returning ships	466	240	1038	1634
No.of unique L_js	101	80	148	169
η_a[%]	3.64	1.78	1.68	1.21

D.2 Ship Size Clustering

The ships were divided into classes (alternatively called clusters) according to the ship lengths L_j. Class i has length range $(c_{i-1}, c_i]$, where $c_0 = 0$. Values of c_i are summarized in Tab. D.2. Also the total number of calls at the port for each cluster and the number of aperiodic arrivals in the cluster are given in the brackets.

In the following discussion we will refer to the clusters using a short-hand notation consisting of the initials of the port name and the cluster number starting from 1 for the shortest ship sizes. For example, RT3 is the third cluster for Rotterdam, with 532 ship calls and in this 2 aperiodic arrivals (cf. Tab.D.2).

Table D.2: Ship size cluster intervals in meters, the total number of calls in the clusters and number of aperiodic calls.

port	Gdańsk	Long Beach	Los Angeles	Le Havre
c_1	137 (115,5)	188 (89,1)	224 (167,9)	140 (216,6)
c_2	151 (103,2)	232 (137,2)	261 (132,4)	210 (337,25)
c_3	183 (121,1)	273 (118,11)	279 (180,5)	245 (333,7)
c_4	210 (11,3)	302 (203,14)	295 (252,12)	278 (400,19)
c_5	300 (17,10)	338 (254,15)	305 (181,12)	300 (353,29)
c_6	368 (50,2)	368 (193,10)	335 (294,13)	368 (528,49)
c_7	399 (54,1)	399 (1,1)	399 (102,16)	399 (104,7)
port	Hamburg	Rotterdam	Shanghai	Singapore
c_1	141 (905,8)	102 (153,2)	101 (642,2)	192 (4552,21)
c_2	170 (956,6)	141 (2395,13)	148 (4144,11)	198 (3206,21)
c_3	213 (298,5)	152 (532,2)	183 (2263,20)	225 (2231,21)
c_4	279 (302,23)	170 (444,4)	237 (1584,30)	262 (2182,47)
c_5	338 (339,45)	223 (240,26)	297 (1800,65)	302 (3188,67)
c_6	369 (422,27)	273 (154,13)	348 (1022,53)	345 (1635,28)
c_7	400 (72,6)	305 (80,11)	367 (151,14)	400 (1500,18)

D.3 p_j/L_j distribution parameters

In this section parameters of the distributions fitting p_j/L_j data in the best way are given. p_j is ship processing time in hours, L_j length in meters for arrival j. `fitdistrplus` R package was used [20]. The following format of result presentation is used: first the cluster short name is given, next the name of the distribution, and the parameters of the distribution. Ship size classes short-hand notation consists of the initials of the port name and the cluster number starting from 1 for the shortest ship sizes. The distribution short names are: β - for beta distribution, exp – for exponential, γ - gamma distribution, no – for normal, ln – lognormal, ls - logistic, We – for Weibull. For more details on distribution parameters see [20].

Gdańsk:
GD1,ln, meanlog -2.2045270, sdlog 0.9806298; GD2,We, shape 1.6085042, scale 0.1253091; GD3,γ, scale 0.02542777, shape 6.13514495; GD4,β, shape1 3.796198, shape2 26.674938; GD5,ln, meanlog -2.383588, sdlog 0.612455; GD6,ln, meanlog -2.0397412 sdlog 0.4411036; GD7,We, shape 6.3138234 scale 0.2000697.

Long Beach:
LB1:ls, location 0.12513149, scale 0.05685868; LB2:We, shape 1.7859668, scale 0.2262769; LB3:ls, location 0.20100499, scale 0.04349006; LB4:ln, meanlog -2.5612589, sdlog 0.9126934; LB5:ls, location 0.23740617 scale 0.07422126; LB6:ls, location 0.23748989 scale 0.02907422; LB7: single item cluster use data directly, p_j=140h L_j=399.2m

Los Angeles:
LA1:ln, meanlog -2.0741789, sdlog 0.3958589; LA2:no, mean 0.12818969, sd 0.05853156; LA3:ls, location 0.1962775, scale 0.0541613; LA4:β,shape1 2.576578, shape2 17.518522; LA5:ls, location 0.23346274,scale 0.03294501; LA6:γ,scale 0.01622799, shape 15.51326663; LA7:ls, location 0.24577252,scale 0.03512682;

Le Havre:
LH1:ln, meanlog -2.4466108, sdlog 0.6849413; LH2:ln, meanlog -2.5382265, sdlog 0.5432081; LH3:ln, meanlog -2.6116072, sdlog 0.4608459; LH4:ls, location 0.05769683, scale 0.01408215; LH5:ln, meanlog -2.944952, sdlog 0.424198; LH6:γ, scale 0.01167714, shape 5.91481882; LH7:γ, scale 0.006849668, shape 11.434991181;

Hamburg:
HB1: We, shape 1.657545, scale 0.187926; HB2: no, mean 0.16878201, sd 0.09806034; HB3: ln, meanlog -2.4404704, sdlog 0.4536681; HB4: ln, meanlog -2.4003098, sdlog 0.3447301; HB5: ls, location 0.10171916, scale 0.01809944; HB6: ln, meanlog -2.2354715, sdlog 0.3202304; HB7: We, shape 3.8331868, scale 0.1257602;

Rotterdam:
RT1: γ, scale 0.05080236, shape 3.24384335; RT2: ln, meanlog -2.264001, sdlog 0.590673; RT3: γ, scale 0.03108345, shape 3.79563199; RT4: γ, scale 0.0293398, shape 3.8016438; RT5: β, shape1 5.195051, shape2 54.868293; RT6: ln, meanlog -2.5495952, sdlog 0.2745269; RT7: ls, location 0.08838709, scale 0.01932081;

Shanghai:
SH1: exp, rate 26.87598; SH2: ls, location 0.08019359, scale 0.01406165; SH3: ls, location 0.06672222, scale 0.01592844; SH4: ls, location 0.05303453, scale 0.01191112; SH5: ln, meanlog -2.9072884, sdlog 0.4474107; SH6: ln, meanlog -2.8731473, sdlog 0.4597978; SH7: ln, meanlog -2.4260391, sdlog 0.9032254;

Singapore:
SI1: γ, scale 0.02191416, shape 4.51531213; SI2: ln, meanlog -2.4243607, sdlog 0.4522818; SI3: γ, scale 0.009468676, shape 7.039842219; SI4: γ, scale 0.01036994, shape 5.90262497; SI5: β, shape1 6.606161, shape2 111.220183; SI6: β, shape1 7.961777, shape2 140.759075; SI7: ls, location 0.0611644, scale 0.0104015;

D.4 Return time ρ_j distributions summary

In this section results of fitting mixture of normal distributions into return times in certain ship size classes are given. R package `normalmixEM` was used. The results are provided in the format: ship size cluster name: ℓ - the number of components in the mixture $(\lambda_1, \mu_1, \sigma_1) \dots (\lambda_\ell, \mu_\ell, \sigma_\ell)$. Time units are days.

There are two special cases: 1. There are only 4 return periods in GD5. Hence, it was not feasible to compute the fitting mixture. Normal distribution was fit using `fitdistrplus` for completeness of presentation and compatibility with the other results, although this fit is disputable and should be considered rather speculative. 2. No results are provided for LB7 because the only ship present – "CMA CGM Benjamin Franklin" – was non-returning.

D.4.1 Gdańsk

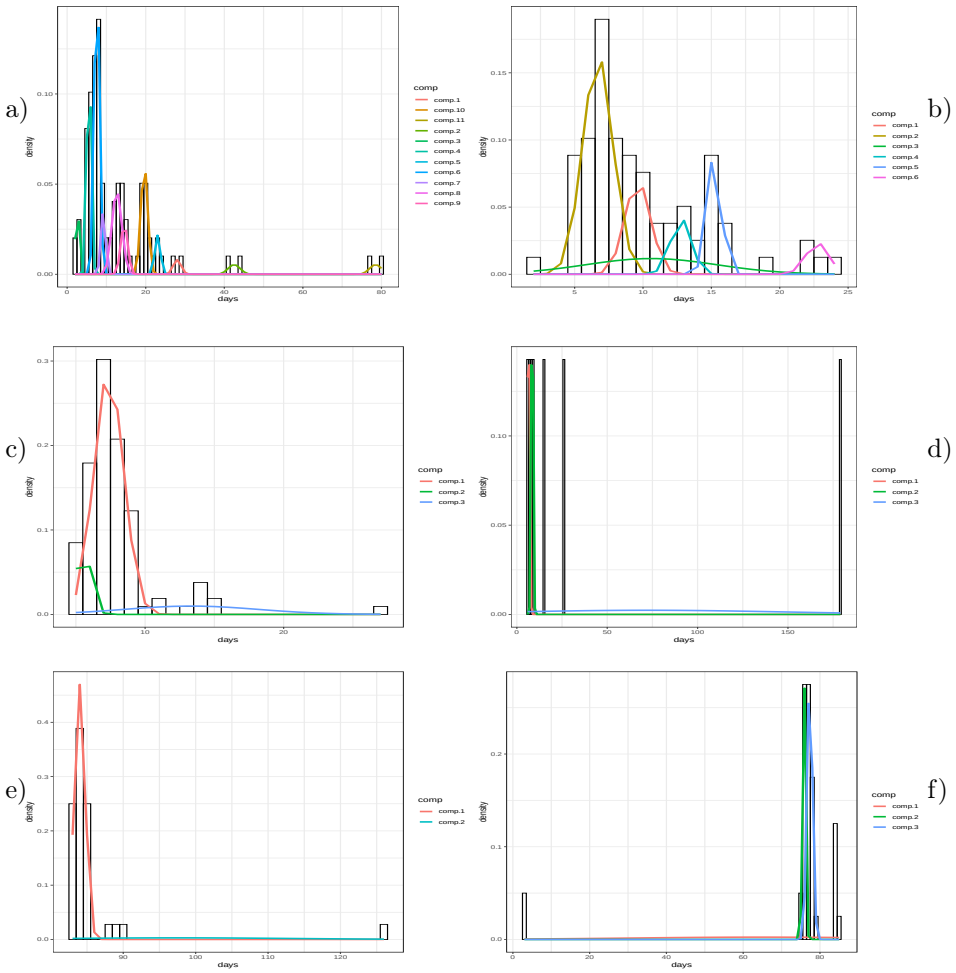


Figure D.1: Return times for Gdańsk clusters, data and the fit normal mixtures. a) GD1 b) GD2 c) GD3 d) GD4 e) GD6 f) GD7.

Gdańsk

GD1: 11, (0.0505, 2.6000, 0.4899), (0.1749, 5.5587, 0.5354), (0.2817, 7.5687, 0.6658), (0.0530, 9.2066, 0.5967), (0.1460, 12.6488, 1.2607), (0.0615, 14.5319, 0.8741), (0.1325, 19.6399, 0.8703), (0.0393, 23.0155, 0.7242), (0.0202, 27.9994, 1.0012), (0.0202, 42.5000, 1.5000), (0.0202, 78.5000, 1.5000)

GD2: 6, (0.4517, 6.7046, 1.1001), (0.1632, 9.6149, 0.9298), (0.1407, 10.6054, 4.7927), (0.0779, 12.7734, 0.7398), (0.1175, 15.2180, 0.5158), (0.0489, 22.7554, 0.8292)

GD3: 3, (0.7653, 7.3724, 1.0533), (0.1153, 5.5130, 0.5313), (0.1194,13.3165, 4.8126)

GD4: 3, (0.2825, 6.5142, 0.5271), (0.2824, 8.4857, 0.5271), (0.4351,72.3471,74.7188)

GD5: 1, (1,27.0000,34.7275)

GD6: 2, (0.8819, 83.9997, 0.7490), (0.1185, 97.4030,15.9130)

GD7: 3, (0.2137, 64.7057, 34.1502), (0.2812, 75.8580, 0.3871), (0.5051, 77.3439, 0.7021)

D.4.2 Hamburg

Hamburg:

Hamburg

HB1: 9, (0.1338, 13.9463, 0.6629) (0.4781, 7.0110, 0.5821) (0.0109, 16.7469, 0.5387) (0.2652, 7.8246, 3.5046) (0.0591, 20.7478, 1.2175) (0.0249, 27.0929, 1.7919) (0.0022, 198.1930, 26.0457) (0.0048, 40.6954, 1.6975) (0.0209, 77.0945, 33.2909)

HB2: 11, (0.1519, 11.2314, 0.5786) (0.0050, 72.9334, 12.0680) (0.2460, 6.9860, 0.8959) (0.0268, 40.7911, 5.2799) (0.0022, 202.5000, 19.5009) (0.0560, 18.8840, 7.3961) (0.0203, 1.3823, 0.7127) (0.2076, 13.9854, 0.9682) (0.0758, 21.3319, 1.1658) (0.2030, 9.7024, 0.6114) (0.0053, 104.4390, 5.9359)

HB3: 11, (0.0370, 15.5176, 0.5581) (0.1119, 13.7237, 0.5692) (0.0394, 21.5831, 0.8858) (0.0474, 38.7644, 0.8111) (0.1521, 35.0377, 0.7766) (0.0284, 69.9998, 0.7520) (0.0950, 15.6744, 6.3291) (0.2923, 42.0441, 0.5804) (0.0205, 44.1639, 0.4428) (0.1527, 55.8268, 1.0967) (0.0233, 78.8618, 33.6429)

HB4: 11, (0.0583, 90.4064, 1.7251) (0.1231, 28.0825, 1.0600) (0.0864, 38.0726, 4.4006) (0.2524, 34.9258, 0.5692) (0.0141, 56.0380, 3.2205) (0.0686, 14.1930, 0.8240) (0.1179, 48.9229, 0.5397) (0.0116, 71.5730, 1.6570) (0.2029, 55.8177, 0.5169) (0.0289, 62.9764, 0.8968) (0.0358, 47.9078, 35.3381)

HB5: 11, (0.0977, 28.0399, 1.1294) (0.0093, 22.5001, 0.5000) (0.1244, 34.8233, 0.5577) (0.0093, 49.0019, 1.0032) (0.0140, 92.6667, 2.3570) (0.0349, 36.7662, 2.6745) (0.0280, 6.8373, 4.2657) (0.0674, 63.9684, 2.9195) (0.5028, 55.7818, 0.9585) (0.0094, 73.5319, 0.5407) (0.1027, 76.7290, 0.7495)

HB6: 11, (0.0307, 56.0110, 1.2115) (0.0534, 63.0652, 0.6720) (0.1348, 69.7168, 0.6585) (0.1842, 69.8943, 1.7559) (0.0506, 75.3359, 0.6348) (0.0323, 93.7556, 4.0808) (0.2138, 77.1603, 0.8522) (0.1204, 83.8510, 0.9491) (0.0037, 80.4333, 0.5943) (0.0266, 89.7338, 0.4925) (0.1494, 105.3820, 28.1660)

HB7: 7, (0.2519, 77.0001, 1.2472) (0.1431, 81.8396, 0.9661) (0.0439, 97.7382, 2.0859) (0.0784, 107.1307, 3.5942)

(0.3910, 89.7361, 1.5419) (0.0429, 84.5313, 0.5143) (0.0488, 150.4860, 10.5242)

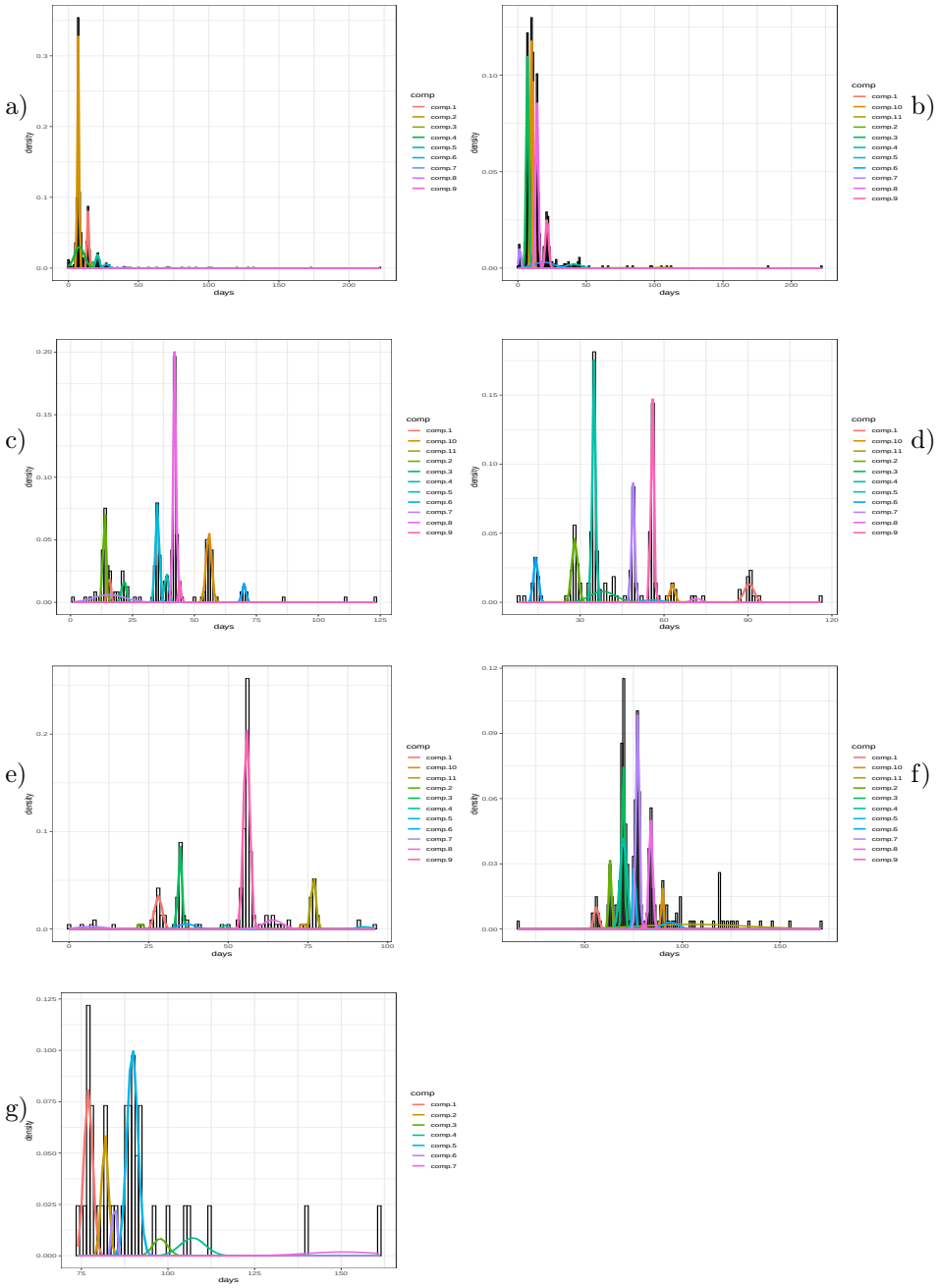


Figure D.2: Return times for Hamburg clusters, data and the fit normal mixtures. a) HB1, b) HB2, c) HB3, d) HB4, e) HB6, f) HB7.

D.4.3 Los Angeles

Los Angeles:

Los Angeles:

LA1: 6, (0.6466, 14.1200, 0.3637) (0.0431, 42.1785, 1.6189) (0.0861, 34.9971, 0.9979) (0.0329, 70.6643, 3.0640) (0.0506, 70.2429, 8.4599) (0.1408, 62.6892, 0.5511)

LA2: 7 (0.0645, 33.8059, 0.4672) (0.0228, 16.9959, 1.4128) (0.3724, 27.9990, 0.6447) (0.1705, 35.1915, 0.4642) (0.0778, 27.7502, 1.9751) (0.2102, 41.9985, 0.3611) (0.0819, 67.2015, 15.6699)

LA3: 6 (0.3129, 14.1283, 0.7010) (0.0128, 46.4996, 0.5000) (0.0821, 67.3920, 4.3001) (0.3131, 42.0392, 0.4649) (0.2458, 35.0270, 0.8514) (0.0333, 76.0867, 30.0495)

LA4: 7 (0.0395, 91.1362, 54.0707) (0.1239, 34.9859, 0.9874) (0.3128, 42.9697, 1.8993) (0.0385, 48.3891, 0.5442) (0.0403, 96.1010, 12.6990) (0.3668, 60.8904, 6.2699) (0.0781, 14.4672, 0.8032)

LA5: 5 (0.5452, 39.6934, 3.3560) (0.0979, 55.8311, 1.3728) (0.0973, 69.9989, 0.3662) (0.1828, 83.6124, 1.2214) (0.0768, 107.5450, 28.8916)

LA6: 6 (0.0177, 1.5000, 1.1180) (0.5321, 42.0079, 0.6314) (0.0088, 178.0000, 10.0000) (0.0098, 65.1838, 3.5678) (0.4007, 45.4147, 4.9146) (0.0309, 89.1863, 15.1729)

LA7: 6 (0.0893, 1.8000, 1.1662) (0.0328, 46.4775, 0.5208) (0.2681, 48.8484, 0.6389) (0.0745, 67.1598, 10.2891)

(0.4464, 116.8400, 3.5177) (0.0889, 42.1978, 0.3984)

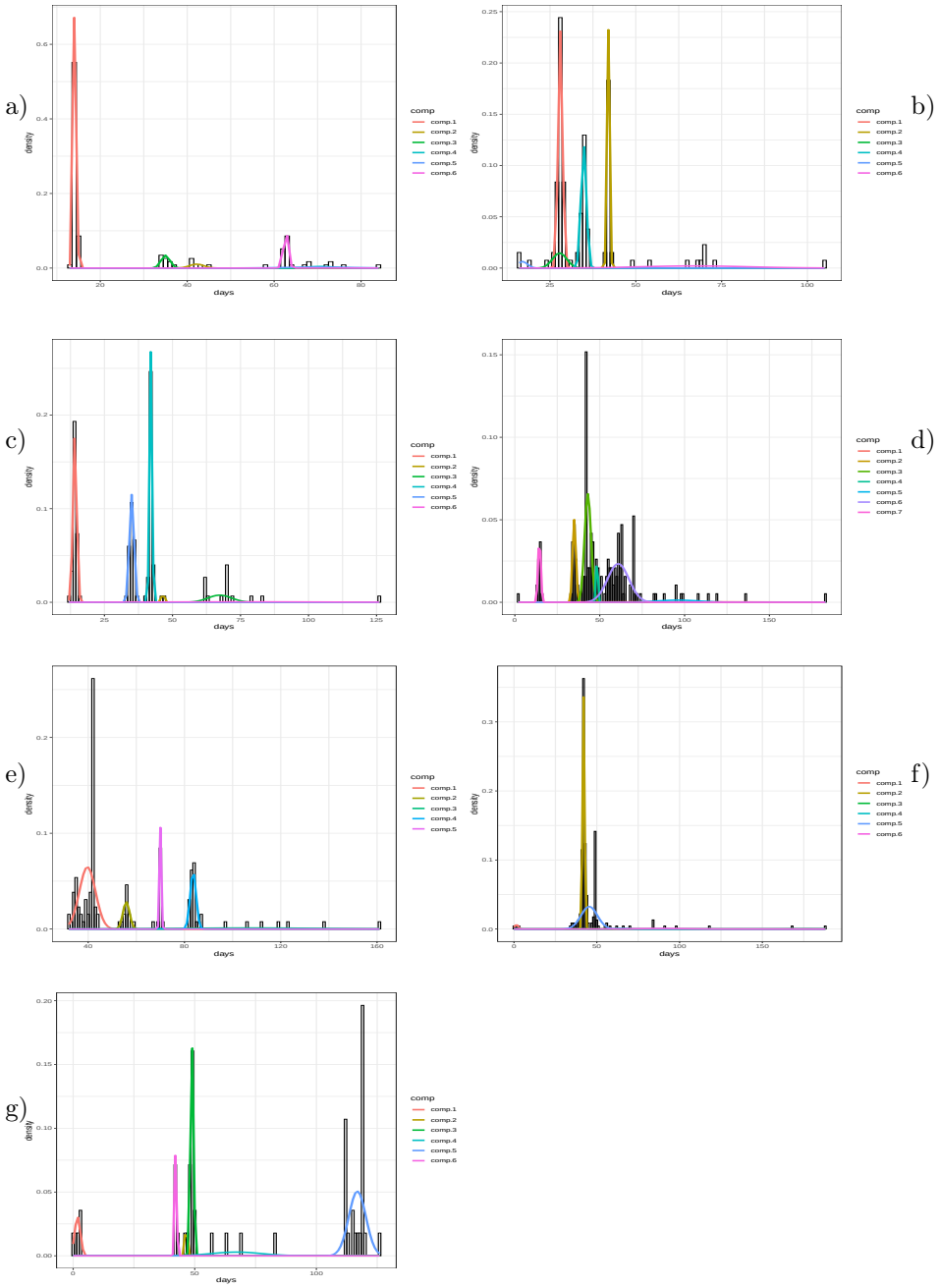


Figure D.3: Return times for Los Angeles clusters, data and the fit normal mixtures. a) LA1, b) LA2, c) LA3, d) LA4, e) LA5, f) LA6, g) LA7.

D.4.4 Long Beach

Long Beach

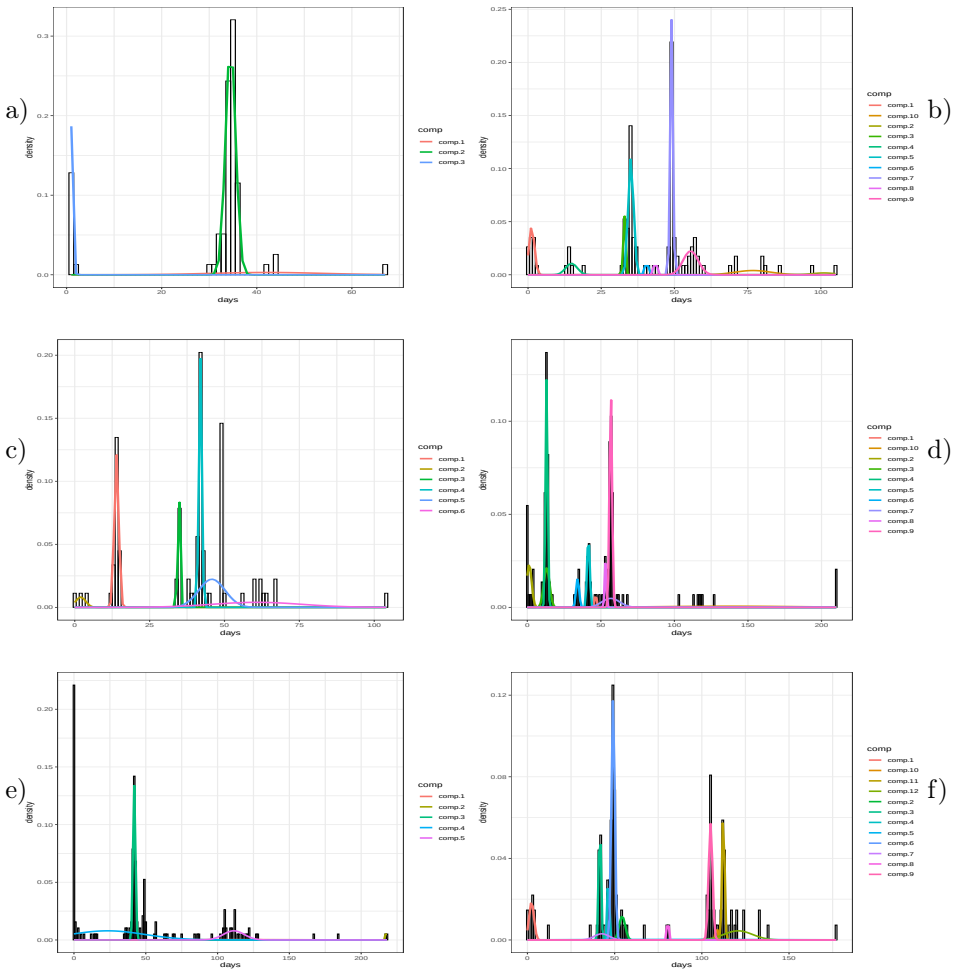


Figure D.4: Return times for Long Beach clusters, data and the fit normal mixtures. a) LB1, b) LB2, c) LB3, d) LB4, e) LB5, f) LB6.

Long Beach:
LB1: 3 (0.1409, 1.0903, 0.2867) (0.7736, 34.4965, 1.0600) (0.0854, 42.2229, 11.8362)
LB2: 10 (0.1053, 1.2500, 0.9242) (0.0174, 101.0275, 4.0156) (0.0546, 32.8600, 0.3674) (0.0526, 14.8333, 1.9508) (0.2524, 35.1154, 0.9184) (0.0175, 40.5000, 0.5000) (0.2768, 49.0235, 0.4592) (0.0175, 43.5000, 0.5000) (0.1440, 55.5140, 2.5589) (0.0618, 76.8213, 6.1814)
LB3: 6 (0.2246, 13.9504, 0.7393) (0.0337, 1.9996, 1.6329) (0.0972, 34.7896, 0.4076) (0.2480, 41.9255, 0.4961) (0.2461, 45.7881, 4.3949) (0.1504, 60.7253, 15.0404)
LB4: 10 (0.0111, 46.4848, 0.4998) (0.0958, 1.2848, 1.6655) (0.1039, 13.1648, 1.9505) (0.2246, 13.1058, 0.7254) (0.0877, 41.5332, 0.9248) (0.0407, 34.1750, 1.0603) (0.0734, 56.7511, 6.0599) (0.0349, 53.1504, 0.5688) (0.2539, 56.7543, 0.8738) (0.0741, 137.5389, 48.1475)
LB5: 5 (0.0422, 105.9515, 50.0724) (0.0105, 217.5001, 0.5000) (0.2904, 41.9595, 0.8649) (0.5130, 23.4697, 25.8211) (0.1439, 111.2706, 7.2218)

LB6: 12 (0.0656, 2.3380, 1.4105) (0.0985, 41.5421, 0.6622) (0.0323, 42.6180, 4.2569) (0.0436, 46.4018, 0.4997)

(0.2848, 49.1267, 0.9613) (0.0504, 54.3180, 1.7206) (0.0142, 80.5000, 0.5000) (0.0377, 87.7966, 58.7436)

(0.1579, 105.0233, 1.1057) (0.0098, 108.5330, 0.5217) (0.1131, 112.3252, 0.7087) (0.0922, 120.8059, 8.2167)

D.4.5 Le Havre

Le Havre

Le Havre:

LH1:6, (0.1151, 4.6301, 0.8343) (0.1740, 15.5915, 2.0145) (0.0685, 21.0844, 1.0550) (0.2201, 36.8575, 22.4692) (0.3928, 42.0277, 0.7392) (0.0296, 201.0784, 89.3193)

LH2: 11, (0.1017, 4.6673, 0.7990) (0.1835, 15.3701, 2.0096) (0.0363, 3.3028, 2.4242) (0.0960, 21.0775, 1.6323) (0.0273, 28.1328, 0.5773) (0.0190, 32.9941, 0.8039) (0.0217, 39.3531, 0.7300) (0.0913, 59.2163, 23.4321) (0.0147, 281.3922, 37.7351) (0.4012, 42.0738, 0.7499) (0.0073, 156.9290, 11.1290)

LH3: 10, (0.1158, 9.9860, 5.4575) (0.0118, 85.5065, 2.0357) (0.2177, 28.1665, 1.1498) (0.0574, 35.2944, 1.1335) (0.1508, 41.9474, 0.7317) (0.2190, 50.5533, 6.8542) (0.0174, 76.3862, 2.1448) (0.1197, 55.9970, 1.0894) (0.0275, 107.5454, 17.8651) (0.0630, 98.0772, 1.2232)

LH4: 19, (0.0504, 13.7125, 0.8502) (0.0054, 17.1766, 0.7615) (0.0359, 25.8101, 0.9190) (0.0268, 11.4005, 6.9706) (0.0292, 31.3876, 1.0657) (0.1722, 28.1633, 0.5894) (0.0100, 37.9291, 0.9321) (0.0063, 77.9617, 2.0447) (0.1523, 35.0051, 0.5868) (0.1066, 41.8168, 0.4591) (0.0294, 95.2060, 5.6932) (0.0229, 43.2141, 0.4999) (0.1108, 49.0712, 0.7752) (0.0171, 46.1998, 1.0419) (0.0803, 55.8154, 0.8558) (0.0175, 62.8218, 2.6557) (0.0782, 90.4310, 0.8564) (0.0357, 105.2259, 1.4236) (0.0133, 165.9805, 27.3269)

LH5: 18, (0.0164, 4.8574, 3.9447) (0.0322, 14.5008, 0.5000) (0.0287, 42.0207, 2.4154) (0.0169, 87.7792, 3.5113) (0.1718, 27.3826, 0.6065) (0.0750, 29.1995, 0.5612) (0.0478, 35.2803, 0.8714) (0.1150, 41.8780, 0.8122) (0.0157, 31.9638, 0.7835) (0.0357, 62.2495, 11.5266) (0.0081, 20.9999, 1.0000) (2.8,E-08 8,0.9138 15.9819) (0.0467, 107.7564, 11.3595) (0.2821, 56.0501, 0.8322) (0.0472, 58.7417, 3.3505) (0.0284, 91.2040, 0.4034) (0.0161, 156.7051, 7.9240) (0.0163, 216.8095, 15.5824)

LH6: 14, (0.1158, 13.2648, 1.1851) (0.0181, 33.3236, 0.7673) (0.0830, 35.1462, 0.3970) (0.0055, 187.9700, 17.0495) (0.1095, 49.0479, 0.5945) (0.1851, 55.9032, 0.6740) (0.0101, 51.9834, 1.3127) (0.0210, 59.0222, 2.1264) (0.0155, 91.8451, 1.1838) (0.0950, 72.2064, 2.0891) (0.0154, 16.6543, 14.5205) (0.1836, 77.0102, 0.6335) (0.0424, 84.8864, 2.0619) (0.0999, 112.5720, 6.7157)

LH7: 11 (0.0758, 11.0000, 2.9665) (0.0791, 71.3722, 1.5090) (0.2348, 77.6234, 0.7853) (0.0948, 75.6707, 1.1326)

(0.0609, 151.1032, 7.3608) (0.1457, 83.9871, 1.6106) (0.0733, 86.2400, 0.9302) (0.0844, 88.5186, 0.5220)

(0.0455, 92.3304, 0.9479) (0.0606, 109.0000, 3.0822) (0.0452, 178.0139, 4.5731)

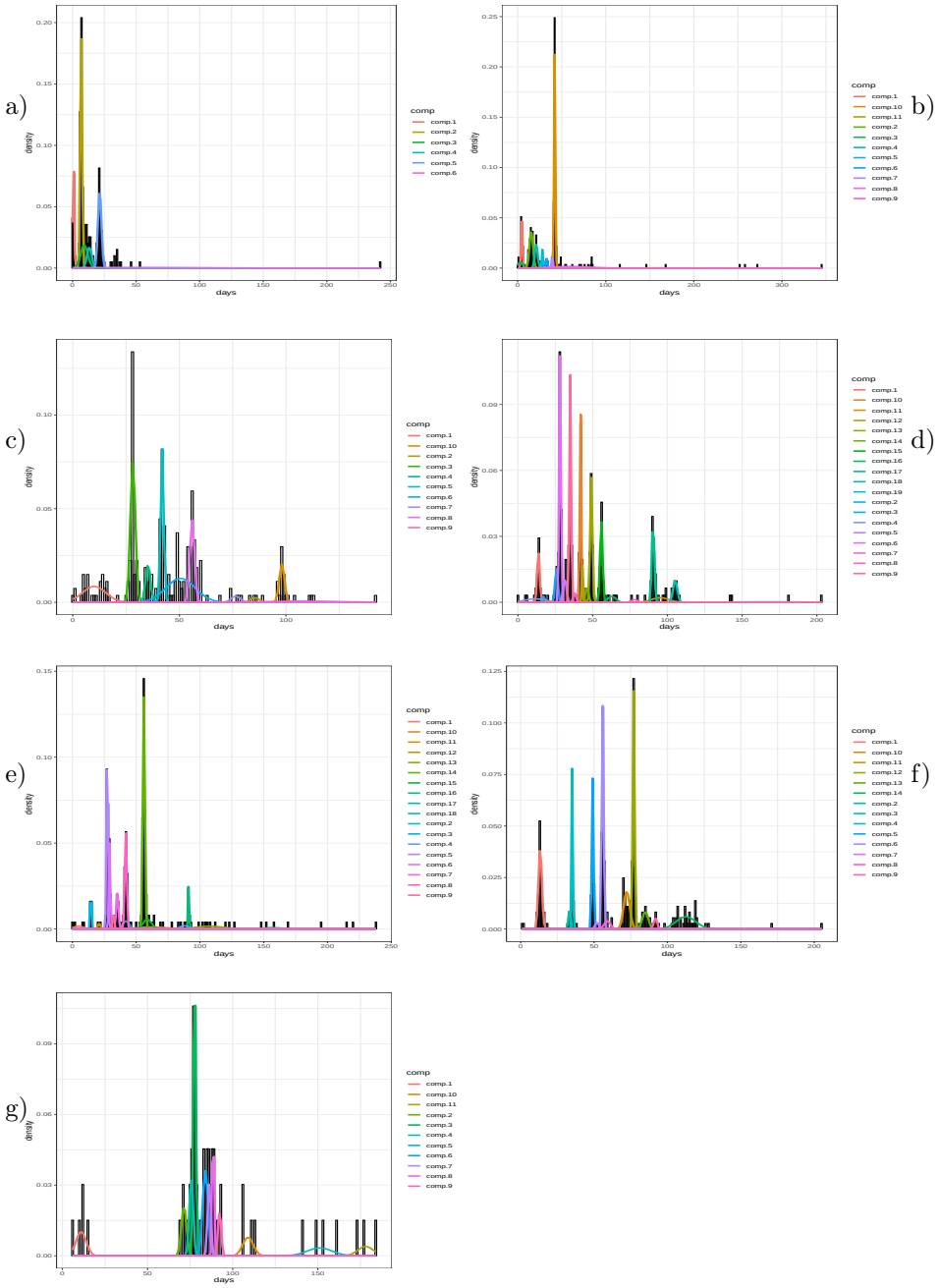


Figure D.5: Return times for Le Havre clusters, data and the fit normal mixtures. a) LH1, b) LH2, c) LH3, d) LH4, e) LH5, f) LH6, g) LH7.

D.4.6 Rotterdam

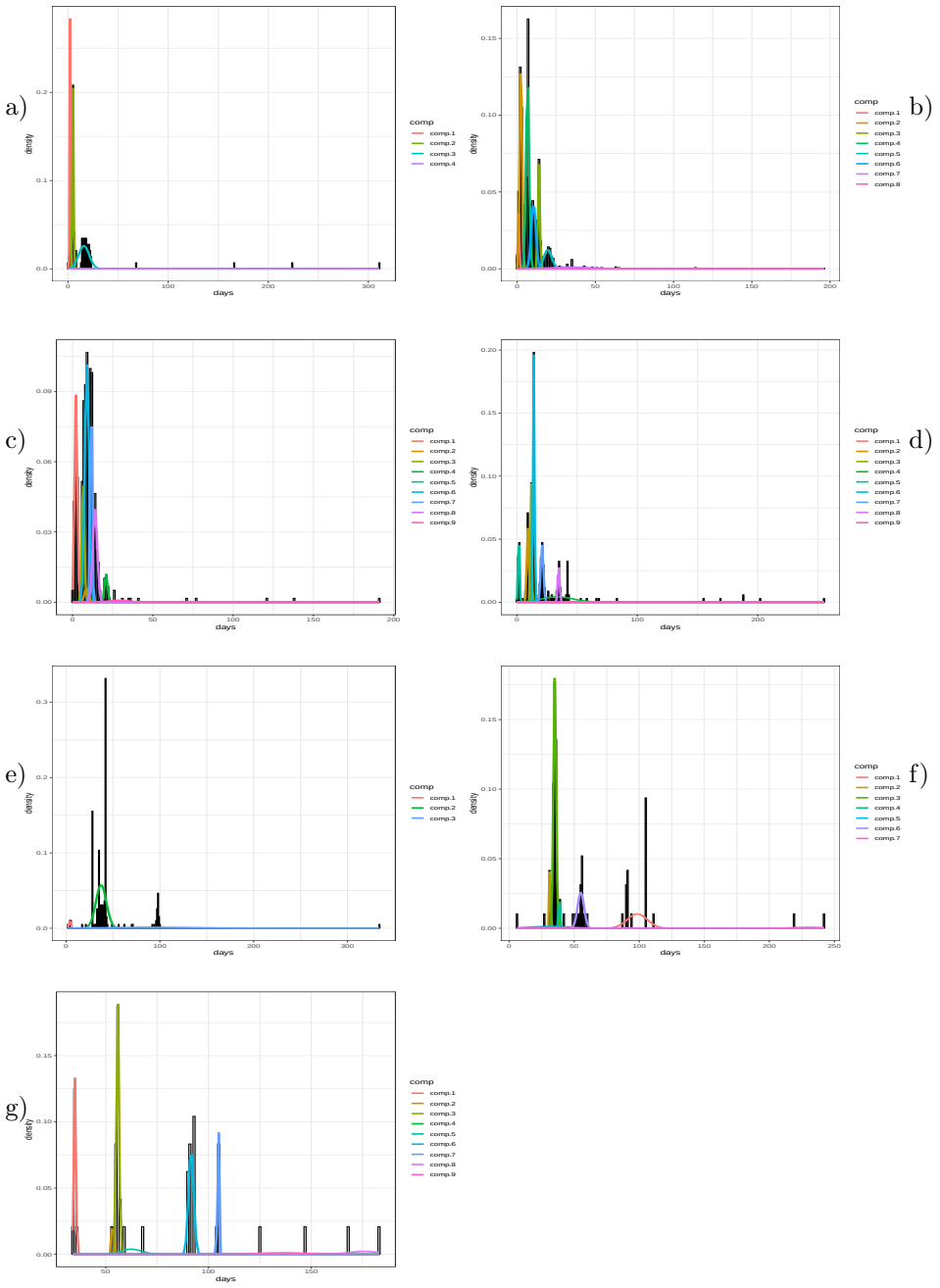


Figure D.6: Return times for Rotterdam clusters, data and the fit normal mixtures. a) RT1, b) RT2, c) RT3, d) RT4, e) RT5, f) RT6, g) RT7.

Rotterdam:

RT1: 4, (0.3219, 2.1212, 0.4361) (0.3059, 4.9244, 0.5922) (0.3435, 15.8448, 5.2419) (0.0287, 186.0621, 93.0158)

RT2: 8, (0.0433, 0.8314, 0.4505) (0.2641, 2.4199, 0.6903) (0.1046, 13.9418, 0.6132) (0.3075, 6.6814, 0.9874) (0.0777, 19.5626, 2.4906) (0.1583, 10.4814, 1.4735) (0.0364, 35.5004, 14.6250) (0.0081, 92.5735, 43.3251)

RT3: 9, (0.1964, 2.0915, 0.8811) (0.0199, 8.7070, 1.3248) (0.1062, 6.5835, 0.7235) (0.0285, 20.8484, 0.9416) (0.0093, 112.9190, 48.5156) (0.3341, 8.9398, 1.3141) (0.1508, 11.5555, 0.6223) (0.1401, 13.9652, 1.4132) (0.0147, 28.9163, 6.9944)

RT4: 9, (0.0158, 158.1524, 65.6611) (0.1242, 9.0092, 0.8485) (0.1379, 11.7955, 0.5474) (0.1429, 33.2861, 13.4660) (0.0877, 1.5672, 0.6069) (0.3030, 13.9432, 0.6140) (0.1263, 20.8740, 1.1102) (0.0556, 34.7185, 0.7639) (0.0065, 191.6630, 6.3088)

RT5: 3, (0.0241, 4.0535, 1.0552) (0.8101, 37.2689, 5.6303) (0.1657, 90.0074, 50.8505)

RT6: 7, (0.1979, 98.4743, 7.7144) (0.0566, 31.2897, 0.4555) (0.4799, 35.0725, 1.0630) (0.0352, 38.5434, 0.5123) (0.0508, 31.4855, 15.2629) (0.1588, 55.0800, 2.4582) (0.0208, 230.5000, 11.5000)

RT7: 9 (0.1667, 35.0000, 0.5000) (0.0375, 53.4756, 0.5204) (0.3095, 55.8453, 0.6352) (0.0095, 136.3048, 11.6007)

(0.0488, 62.3425, 5.0421) (0.2500, 91.5833, 1.2555) (0.1041, 104.8003, 0.3998) (0.0412, 175.5725, 7.5442)

(0.0327, 136.3576, 11.6067)

D.4.7 Shanghai

Shanghai:

Shanghai:

SH1: 2, (0.6331, 1.6613, 1.0242) (0.3669, 14.1485, 13.4295)

SH2: 8, (0.0314, 0.3160, 0.4766) (0.4881, 6.9727, 0.5419) (0.1302, 14.3095, 1.3639) (0.0141, 34.2408, 5.5631) (0.2757, 8.4291, 2.4241) (0.0068, 62.7254, 29.4980) (0.0517, 20.2059, 1.7175) (0.0020, 219.8740, 12.0298)

SH3: 13, (0.0261, 0.5523, 0.6736) (0.0074, 162.5770, 35.1967) (0.1235, 8.9003, 2.1304) (0.2087, 7.1334, 0.6618) (0.2537, 20.7196, 4.1843) (0.0694, 13.9479, 0.7193) (0.0211, 27.5848, 0.6058) (0.0370, 31.6271, 2.7398) (0.0310, 42.1329, 2.6936) (0.1634, 21.2666, 1.0828) (0.0119, 33.3854, 0.5469) (0.0231, 41.6914, 0.5538) (0.0238, 55.8583, 20.6266)

SH4: 11, (0.0183, 0.1496, 0.3567) (0.1515, 6.8757, 1.9607) (0.1386, 14.4486, 1.3933) (0.2740, 21.5482, 2.0014) (0.0694, 26.9064, 0.8405) (0.0848, 28.3127, 0.5740) (0.0830, 32.7388, 3.5934) (0.0572, 35.1291, 0.5930) (0.0305, 40.2626, 1.8842) (0.0815, 60.0605, 14.7204) (0.0113, 145.2768, 68.3605)

SH5: 11, (0.0323, 63.0913, 1.0583) (0.0226, 28.2338, 0.9268) (0.0837, 35.0308, 0.7697) (0.0193, 0.4434, 0.5042) (0.3840, 41.8215, 0.7085) (0.0299, 83.9540, 0.7978) (0.0201, 144.7515, 60.2833) (0.0242, 6.4842, 2.8809) (0.0680, 21.6077, 7.5080) (0.2289, 43.1780, 5.9113) (0.0871, 57.0652, 17.9010)

SH6: 12, (0.0624, 4.9412, 4.0414) (0.1667, 51.6253, 11.7386) (0.1244, 41.1000, 2.5188) (0.0163, 198.8757, 44.0337) (0.0299, 18.9097, 2.9557) (0.2408, 41.9505, 0.8072) (0.0441, 35.2652, 0.5337) (0.0368, 125.6669, 0.9901) (0.1351, 48.7999, 1.0504) (0.0676, 55.9205, 0.5994) (0.0286, 83.3486, 1.3103) (0.0472, 108.2039, 10.6441)

SH7: 7, (0.0305, 14.6687, 0.4707) (0.3566, 3.4906, 2.4873) (0.0447, 9.6400, 0.4917) (0.0328, 43.3592, 15.1970)

(0.2515, 47.9929, 5.9499) (0.1182, 126.8345, 43.5734) (0.1658, 111.7361, 0.9159)

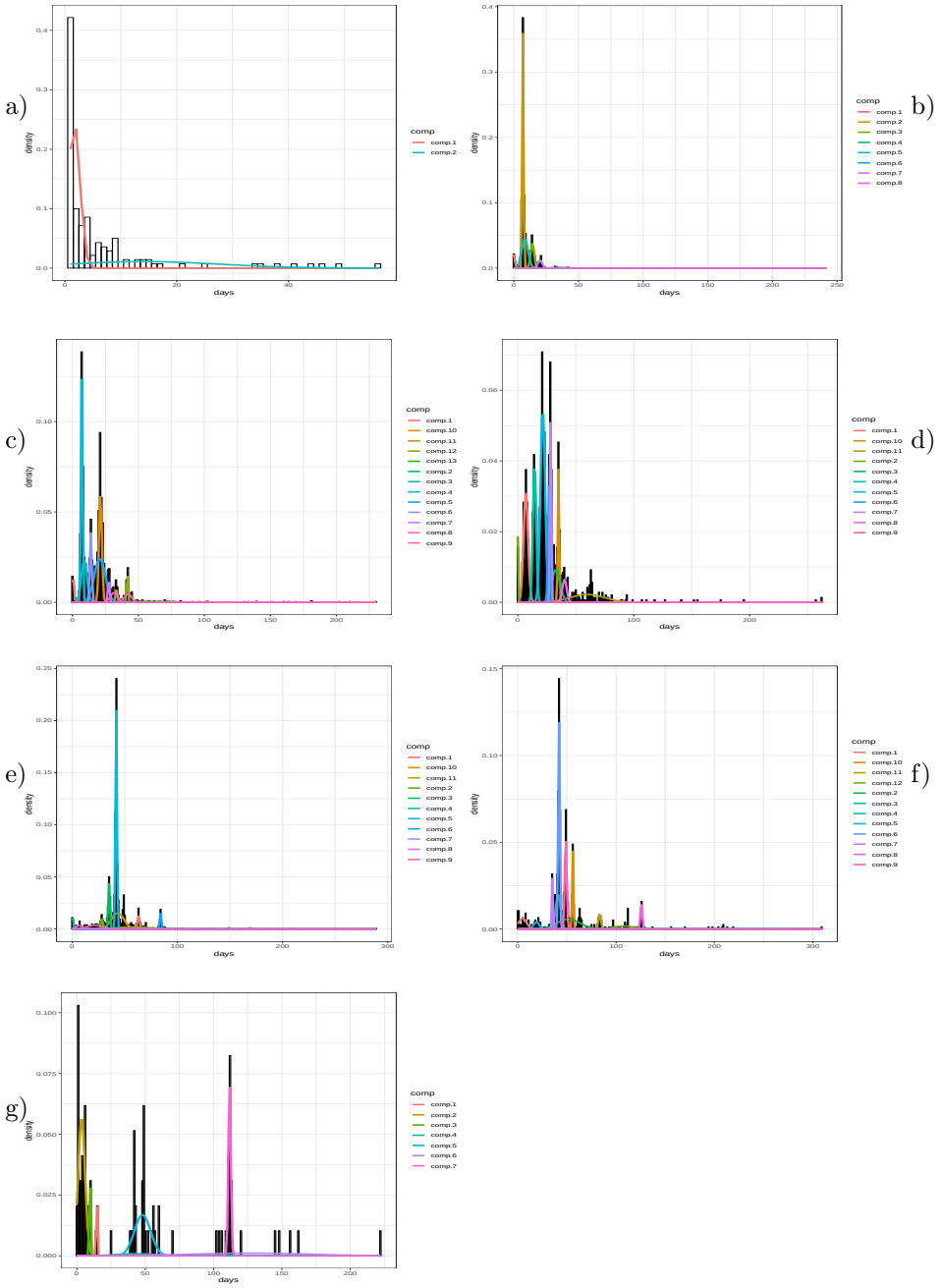


Figure D.7: Return times for Shanghai clusters, data and the fit normal mixtures. a) SH1, b) SH2, c) SH3, d) SH4, e) SH5, f) SH6, g) SH7.

D.4.8 Singapore

Singapore:

Singapore:

SI1: 6, (0.0953, 1.4373, 0.8761) (0.2666, 7.0003, 0.5380) (0.5906, 13.6287, 5.2837) (0.0095, 30.3870, 2.0189) (0.0294, 45.9045, 16.0767) (0.0088, 127.48, 66.4079)

SI2: 4, (0.5687, 4.1500, 3.2873) (0.1437, 14.0320, 0.5509) (0.2700, 21.3500, 11.6884) (0.0177, 74.3963, 52.1076)

SI3: 9, (0.0651, 0.8983, 0.3278) (0.0762, 13.6129, 0.5728) (0.0015, 323.6830, 10.2724) (0.3759, 6.5573, 1.6881) (0.0319, 10.3550, 1.0120) (0.0073, 125.9380, 54.5762) (0.3296, 25.6219, 4.4997) (0.1016, 43.8550, 6.7602) (0.0110, 64.6700, 11.0786)

SI4: 9, (0.0554, 13.2004, 1.2091) (0.3031, 3.7077, 2.8289) (0.0203, 16.3365, 0.5390) (0.0319, 31.1633, 0.9176) (0.2646, 21.8144, 2.4491) (0.0742, 27.7924, 0.7819) (0.0473, 34.3067, 0.8375) (0.1936, 50.7104, 18.2230) (0.0095, 185.5300, 78.8977)

SI5: 10, (0.2003, 2.9135, 2.5237) (0.0394, 41.8823, 1.2544) (0.0126, 162.1931, 62.0449) (0.0287, 18.7046, 0.6496) (0.0421, 20.7642, 0.7104) (0.3930, 20.4949, 7.0738) (0.0816, 22.3735, 0.9400) (0.0535, 35.3515, 1.1482) (0.1159, 61.9371, 9.3212) (0.0327, 85.9281, 15.4469)

SI6: 14, (0.0146, 183.9882, 38.6463) (0.0335, 0.7790, 0.5456) (0.0103, 6.5736, 0.5106) (0.0672, 22.9341, 9.4418) (0.2835, 20.9381, 4.6111) (0.1574, 25.1138, 1.2747) (0.0245, 37.0717, 1.1476) (0.0210, 29.7434, 0.7374) (0.0121, 86.8867, 5.2747) (0.1304, 49.0826, 3.3492) (0.0373, 56.2973, 0.9421) (0.0096, 77.2687, 0.4433) (0.1712, 65.8876, 3.6990) (0.0274, 111.8411, 11.3312)

SI7: 20 (0.1502, 62.6794, 4.3268) (0.1068, 26.5205, 1.5808) (0.1001, 45.5012, 1.2485) (0.0981, 49.0512, 0.7341) (0.0726, 23.3393, 0.7232) (0.0711, 76.4646, 1.0546) (0.0699, 16.9603, 1.2271) (0.0620, 53.2550, 2.0498) (0.0590, 21.3225, 0.9806) (0.0587, 110.6465, 11.0962) (0.0304, 84.3472, 1.2408) (0.0288, 53.3357, 0.6254) (0.0200, 28.0137, 10.6619) (0.0184, 84.3330, 4.3330) (0.0138, 0.4702, 0.4991) (0.0106, 153.5931, 30.6550) (0.0101, 29.9137, 0.7820) (0.0089, 79.4122, 0.5620) (0.0067, 38.2505, 0.5259) (0.0039, 34.8108, 0.5357)

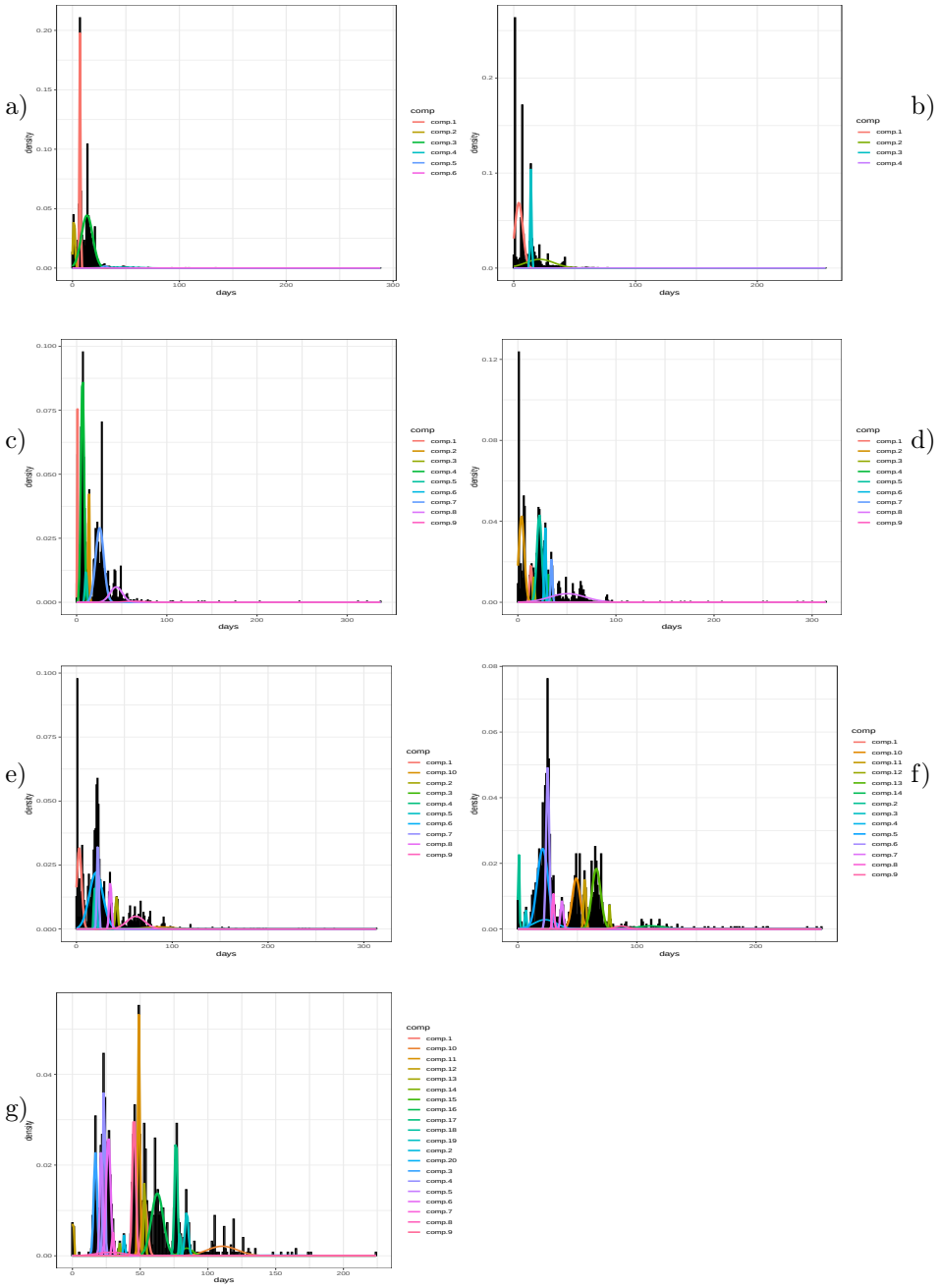
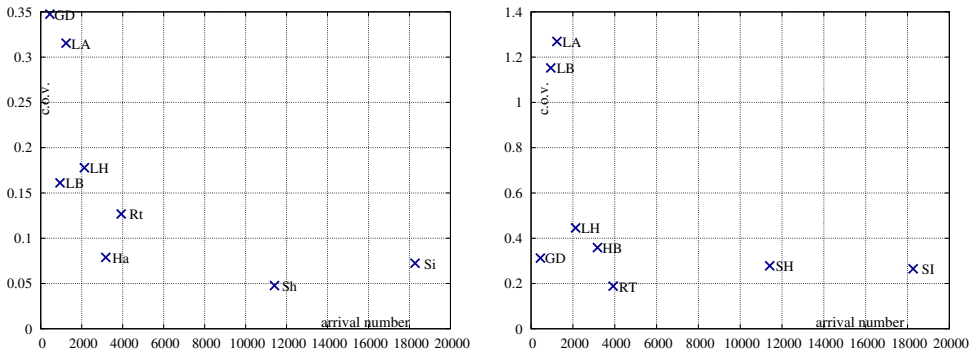


Figure D.8: Return times for Singapore clusters, data and the fit normal mixtures. a) SI1, b) SI2, c) SI3, d) SI4, e) SI5, f) SI6, g) SI7.

D.5 Periodic arrivals

In this section we provide histograms of periodic returning ships arrival collected over all classes. These can be used to generate the first arrival of a returning ship. Arrivals frequencies are shown as fractions of all periodic arrivals. Ship size classes are not distinguished. In Fig.D.9 coefficients of variation of ships weekly and daily arrivals are shown vs total number of arrivals in a port. It can be seen that variability of arrivals decreases with the number of served calls.



a) b)

Figure D.9: Coefficients of variation for periodic arrivals frequencies a) over days of week, b) over hours of day.

D.5.1 Histograms for Days of Week periodic arrivals

D.5.2 Histograms for Hours of Day periodic arrivals

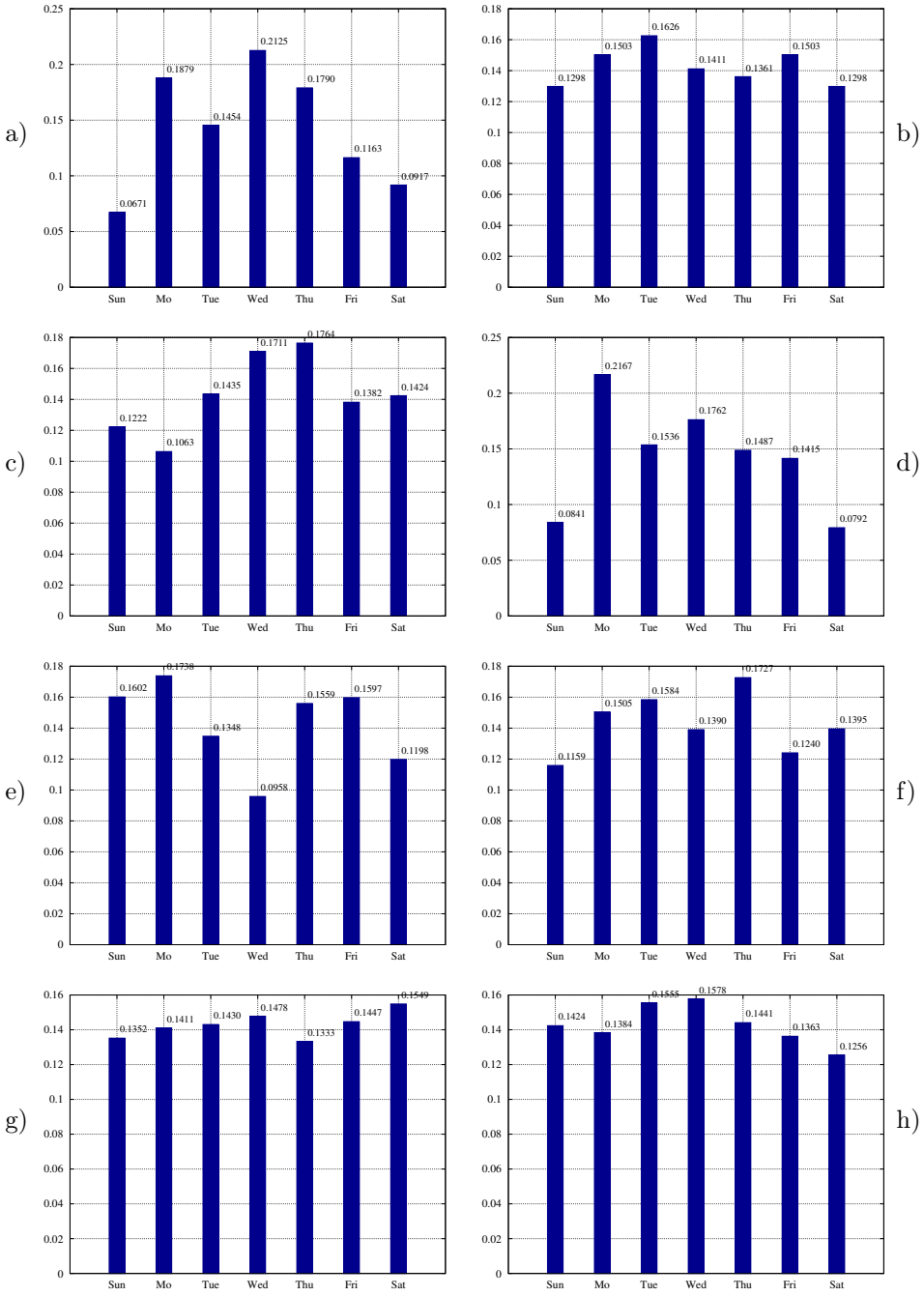


Figure D.10: Histograms of periodic ship arrivals frequencies (relative to all periodic arrivals) over days of week. All classes together. a) Gdańsk, b) Hamburg c) Long Beach, d) Los Angeles, e) Le Havre. f) Rotterdam g) Shanghai h) Singapore.

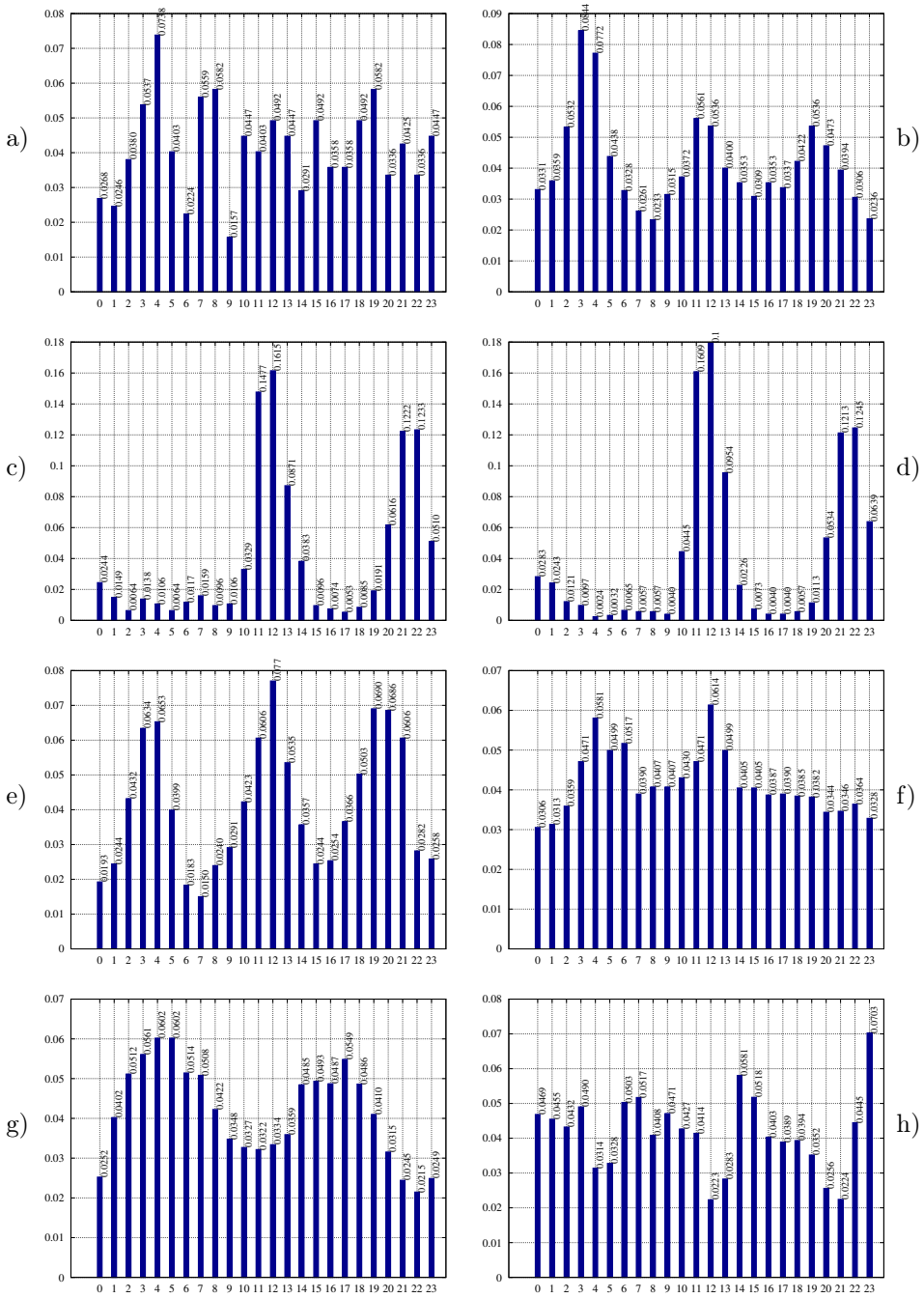


Figure D.11: Histograms of periodic ship arrivals frequencies (relative to all periodic arrivals) over hours of day. All classes together. a) Gdańsk, b) Hamburg c) Long Beach, d) Los Angeles, e) Le Havre. f) Rotterdam g) Shanghai h) Singapore.

D.6 Aperiodic ready times

D.6.1 Weeks of the year

In this section coefficients $a_4, \ldots, a_0$ of the 4th degree polynomials $f(w) = a_4 w^4 + a_3 w^3 + a_2 w^2 + a_1 w^1 + a_0$, where w is the number of a week in year, fit into relative weekly arrival frequencies of aperiodic ships are provided. Ship size classes are not distinguished, i.e. all classes were collected in the input data. These polynomials can be used as a model of probability density of aperiodic arrivals in certain week w. The coefficients are provided in the order $(a_0, a_1, a_2, a_3, a_4)$. Results for the sum of all port weekly aperiodic ships arrival frequencies are also provided. Coefficients of lower degree polynomials are also provided.

degree 4:
Gdańsk: (3.539E-02, -3.249E-03, 2.313E-04, -6.265E-06, 5.474E-08)
Hamburg: (7.484E-02, -9.514E-03, 5.526E-04, -1.298E-05, 1.029E-07)
Long Beach: (8.246E-03, 4.799E-03, -4.873E-04, 1.584E-05, -1.582E-07)
Los Angeles: (1.237E-02, 3.044E-03, -2.483E-04, 6.803E-06, -6.012E-08)
Le Havre: (5.085E-02, -1.837E-03, -4.182E-05, 2.442E-06, -2.206E-08)
Rotterdam: (5.488E-02, -3.038E-03, -1.281E-05, 2.973E-06, -3.418E-08)
Shanghai: (5.128E-02, -6.546E-03, 4.039E-04, -1.018E-05, 8.994E-08)
Singapore: (6.248E-02, -8.546E-03, 5.044E-04, -1.277E-05, 1.178E-07)
Sum of all ports: (4.379E-02, -3.111E-03, 1.127E-04, -1.767E-06, 1.135E-08)

degree 3:
Gdańsk: (2.798E-02, -7.153E-04, 2.474E-05, -3.527E-07)
Hamburg: (6.090E-02, -4.752E-03, 1.644E-04, -1.864E-06)
Long Beach: (2.967E-02, -2.523E-03, 1.097E-04, -1.251E-06)
Los Angeles: (2.051E-02, 2.619E-04, -2.150E-05, 3.103E-07)
Le Havre: (5.384E-02, -2.858E-03, 4.143E-05, 5.952E-08)
Rotterdam: (5.950E-02, -4.619E-03, 1.161E-04, -7.184E-07)
Shanghai: (3.910E-02, -2.383E-03, 6.451E-05, -4.660E-07)
Singapore: (4.652E-02, -3.093E-03, 5.985E-05, -4.595E-08)
Sum of all ports: (4.225E-02, -2.585E-03, 6.991E-05, -5.410E-07)

degree 2:
Gdańsk: (2.505E-02, -9.236E-05, -3.827E-06)
Hamburg: (4.540E-02, -1.460E-03, 1.339E-05)
Long Beach: (1.927E-02, -3.143E-04, 8.392E-06)
Los Angeles: (2.309E-02, -2.862E-04, 3.638E-06)
Le Havre: (5.433E-02, -2.963E-03, 4.625E-05)
Rotterdam: (5.353E-02, -3.350E-03, 5.794E-05)
Shanghai: (3.522E-02, -1.560E-03, 2.677E-05)
Singapore: (4.613E-02, -3.012E-03, 5.613E-05)
Sum of all ports: (3.775E-02, -1.630E-03, 2.609E-05)

D.6.2 Aperiodic arrivals over days of a week (DoW)

Histograms

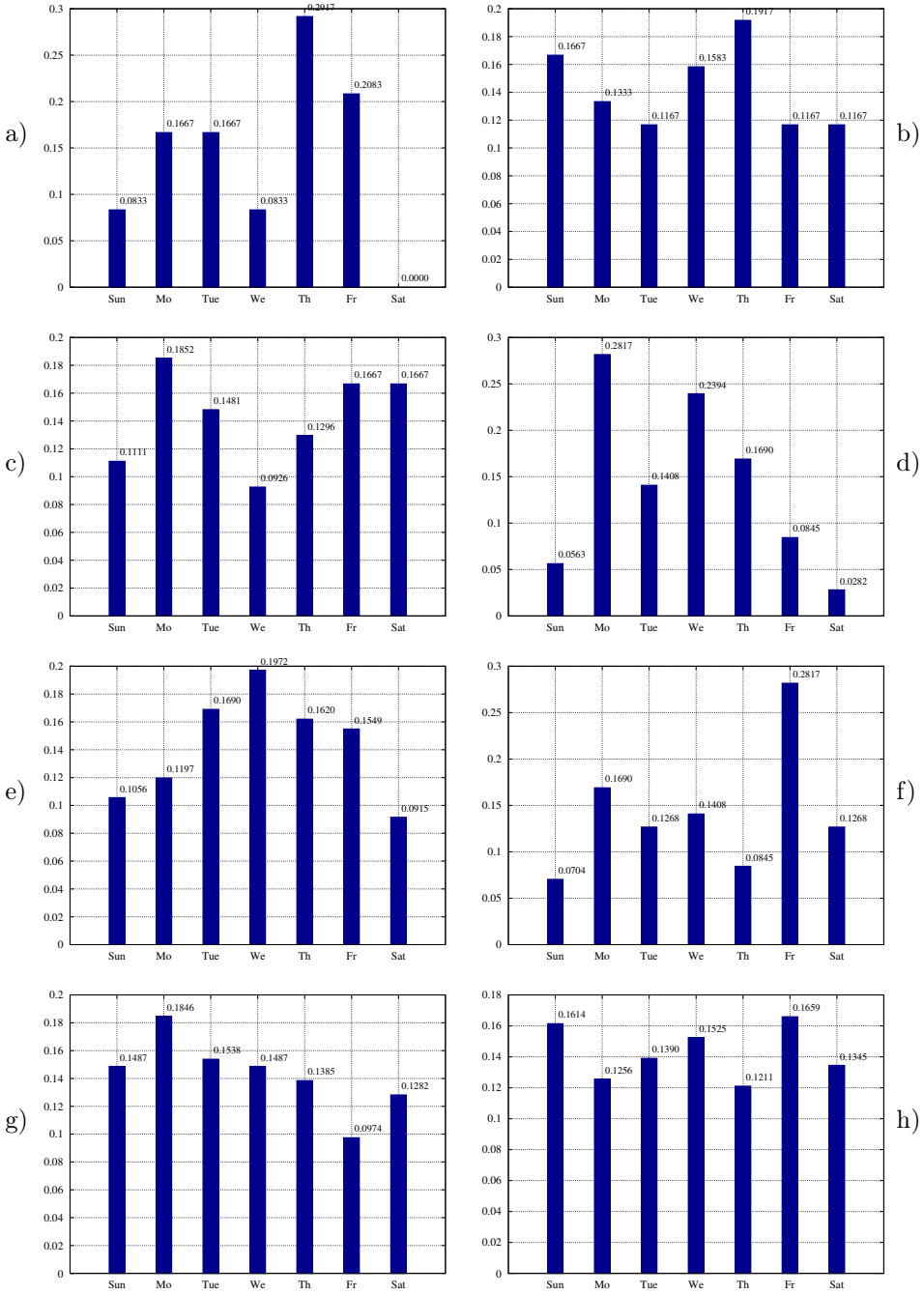


Figure D.12: Histograms of aperiodic ship arrivals frequencies (relative to all aperiodic arrivals) over days of week. All classes together. a) Gdańsk, b) Hamburg c) Long Beach, d) Los Angeles, e) Le Havre. f) Rotterdam g) Shanghai h) Singapore.

Fitting distributions

In this section we provide parameters of the best distributions fitting the fractions of total aperiodic arrivals over days of week. Ship size classes are not distinguished. The reader should be aware that the distributions are day of week-agnostic. This means that the fractions of daily arrivals are treated as a set of values not being aware that there is a sequence of days in a week. The best distribution was chosen on the basis of Anderson-Darling statistic. Entry "All ports" represents results aggregated over all ports. The reader should be also aware that the obtained distributions, except for "All ports", are calculated on just 7 data points and quality of fit is low. With so little data reliable fit of continuous distributions may be unwarranted. In general, we recommend using "All ports" distributions. The distribution short names are: γ - gamma distribution, ls - logistic, We – for Weibull. For more details on distribution parameters see [20].

Days of a week

Gdańsk: ls, location 0.14253365, scale 0.05230696;
Long Beach: We, shape 5.5613990, scale 0.1551807;
Los Angeles: We, shape 1.661107, scale 0.159939;
Le Havre: We, shape 4.6678276, scale 0.1566369;
Hamburg: We, shape 5.493224, scale 0.154562;
Rotterdam: γ, scale 0.02531965, shape 5.64210113;
Shanghai: ls, location 0.14362599, scale 0.01355827;
Singapore: ls, location 0.1424664, scale 0.0100078;

All ports: ls, location 0.14130991, scale 0.02846178;

D.6.3 Aperiodic arrivals over hours of a day (HoD)

Histograms

Fitting distributions

In this section distributions fit to aperiodic arrivals accumulated over hours of a day are shown. Ship size classes are not distinguished. The reader should be aware that the distributions are time-agnostic. This means that the fractions of hourly arrivals are treated as a set of values while ignoring that there is a sequence of hours in time. Still, we decided to provide even these results for completeness. The reader should also be aware that there are few data points and distribution fit is disputable. Consequently, there are cases which are best according to Anderson-Darling statistic but still do not pass Anderson-Darling `gofstat` test as defined in this function of `fitdistrplus` package R. Such results are marked with a star ("*"). The distributions that passed (actually were not rejected by) the `gofstat` test were preferred over the failing ones. For more details on distribution parameters see [20]. The notation is the same as in Section D.6.2.

Hours of day

Gdańsk: ls, location 0.03339101, scale 0.02813923; *
Long Beach: ls, location 0.03232727, scale 0.02567877; *
Los Angeles: ls, location 0.03124656, scale 0.02862704; *
Le Havre: ls, location 0.03835805, scale 0.01687393;
Hamburg: γ, scale 0.009688798, shape 4.300531079;
Rotterdam: ls, location 0.03880830, scale 0.01769015; *
Shanghai: γ, scale 0.0057714, shape 7.2195753;
Singapore: ls, location 0.04201991, scale 0.01170694;

All ports: γ, scale 0.005002241, shape 8.329543747;

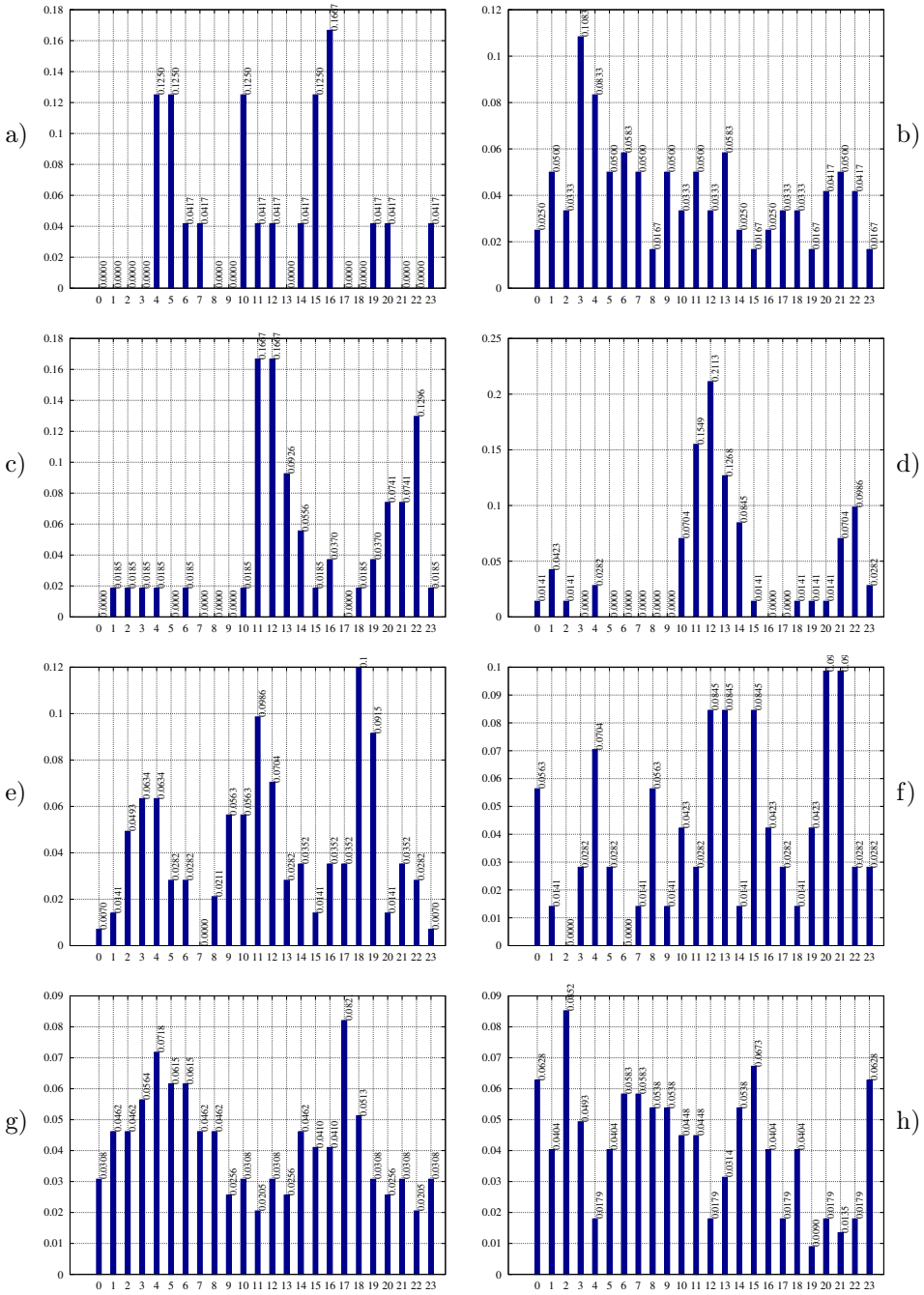


Figure D.13: Histograms of aperiodic ship arrivals frequencies (relative to all aperiodic arrivals) over hours of day. All classes together. a) Gdańsk, b) Hamburg c) Long Beach, d) Los Angeles, e) Le Havre. f) Rotterdam g) Shanghai h) Singapore.

D.7 L_j distributions in clusters

This section contains the summary of the parameters of the distributions fitting ship lengths L_j in a certain cluster of a port. These distributions may be applied as alternative to rounding ships lengths to the upper end of cluster range sizes. A reader should be aware that in certain clusters no distribution can be fit reliably due to scarcity of data or narrow range of values. Cases where distribution fitting procedures failed are marked with '*'. In such cases the upper end of class sizes must be used. The best distribution was chosen on the basis of Anderson-Darling statistic. There is no distribution fit in class LB7 which had only one aperiodic ship. The following format of result presentation is used: first the cluster short name is given, next the name of the distribution, and the parameters of the distribution. The distribution short names are: β - for beta distribution, exp – for exponential, γ - gamma distribution, no – for normal, ln – lognormal, ls - logistic, We – for Weibull. Then, the name is followed by the distribution parameters. For more details on distribution parameters see [20]. Since β distribution is defined in interval [0,1], ship lengths were scaled to the upper end of the cluster, i.e., value L_j/c_i is representing ship j in cluster $(c_{i-1}, c_i]$ of ship lengths.

Gdańsk:
GD1, ln, meanlog 4.8539, sdlog 0.0681; GD2, * ; GD3, We, shape 36.0802, scale 179.5612; GD4, * ; GD5, ls, location 284.3670, scale 10.6077; GD6, * ; GD7, *.

Hamburg:
HB1, We, shape 15.5858, scale 132.6229; HB2, ls, location 160.1010, scale 4.7804; HB3, ls, location 198.6465, scale 8.8947; HB4, We, shape 18.2585, scale 266.5379; HB5, ls, location 312.8788, scale 12.5874; HB6, ls, location 366.2584, scale 0.8339; HB7, We, shape 388.6443, scale 399.0706.

Long Beach:
LB1, ls, location 179.4925, scale 7.9933; LB2, ls, location 217.9850, scale 5.3788; LB3, β, shape1 44.5002, shape2 2.3694; LB4, β, shape1 46.3582, shape2 1.6825; LB5, β, shape1 104.1087, shape2 1.3893; LB6, ls, location 356.7786, scale 5.3755; LB7, *

Los Angeles:
LA1, We, shape 21.2431, scale 211.1785; LA2, ls, location 253.9361, scale 3.9852; LA3, β, shape1 125.4677, shape2 2.7138; LA4, β, shape1 99.9067, shape2 0.9868; LA5, ls, location 301.0938, scale 1.0893; LA6, ls, location 331.1834, scale 3.9768; LA7, We, shape 34.3693, scale 365.7491.

Le Havre:
LH1, meanlog 4.9146, sdlog 0.0282; LH2, We, shape 17.4184, scale 197.1736; LH3, ls, location 231.7942, scale 5.7629; LH4, β, shape1 31.3132, shape2 1.0046; LH5, We, shape 72.47073, scale 296.1866; LH6, ls, location 347.8790, scale 11.9925; LH7, ls, location 397.7178, scale 0.8378;

Rotterdam:
RT1, *; RT2, We, shape 47.7901, scale 139.2240, RT3, *; RT4, ls, location 160.8292, scale 3.2152; RT5, ls, location 207.4839, scale 8.4146; RT6, β, shape1 22.3510, shape2 1.4234; RT7, location 284.7580, scale 5.6919;

Shanghai:
SH1, *; SH2, We, shape 27.4311, scale 143.6816; SH3, We, shape 26.5179, scale 172.4476; SH4, β, shape1 24.4726, shape2 2.9468; SH5, ln, meanlog 5.5995, sdlog 0.0487; SH6, β, shape1 30.8465, shape2 2.6373; SH7, ls, location 356.5223, scale 5.2851.

Singapore:
SI1, We, shape 11.83698, scale 159.75541; SI2, no, mean 185.016904, sd 6.311326; SI3, ls, location 212.8532, scale 4.7380; SI4, We, shape 36.6332, scale 257.5605; SI5, ls, location 282.1924, scale 8.2234; SI6, β, shape1 51.9565, shape2 2.7985; SI7, ls, location 365.5421, scale 6.9683.

Bibliography

[1] Akash Gupta, Debjit Roy, R. d. K., and Parhi, S. Optimal stack layout in a sea container terminal with automated lifting vehicles. *International Journal of Production Research 55*, 13 (2017), 3747–3765.

[2] Asperen, E., Dekker, R., Polman, M., and Arons, H. Modeling ship arrivals in ports. In *Proceedings of the 2003 Winter Simulation Conference* (01 2004), pp. 1737 – 1744 vol.2.

[3] Beasley, J. OR-Library. http://people.brunel.ac.uk/~mastjjb/jeb/info.html, 2018. Accessed: 6 Aug. 2024.

[4] Bellsolà Olba, X., Daamen, W., Vellinga, T., and Hoogendoorn, S. Network capacity estimation of vessel traffic: An approach for port planning. *Journal of Waterway Port Coastal and Ocean Engineering 143* (09 2017).

[5] Bellsolà Olba, X., Daamen, W., Vellinga, T., and Hoogendoorn, S. P. State-of-the-art of port simulation models for risk and capacity assessment based on the vessel navigational behaviour through the nautical infrastructure. *Journal of Traffic and Transportation Engineering (English Edition) 5*, 5 (2018), 335–347.

[6] Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. S. mixtools: An R package for analyzing mixture models. *Journal of Statistical Software 32*, 6 (2009), 1–29.

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

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

[9] Błądek, I., Drozdowski, M., Guinand, F., and Schepler, X. On contiguous and non-contiguous parallel task scheduling. *Journal of Scheduling 18*, 5 (2015), 487–495.

[10] Bouzekri, H., Alpan, G., and Giard, V. Integrated laycan and berth allocation and time-invariant quay crane assignment problem in tidal ports with multiple quays. *European Journal of Operational Research 293*, 3 (2021), 892–909.

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

[12] Byung Kwon Lee, L. H. L., and Chew, E. P. Analysis on high throughput layout of container yards. *International Journal of Production Research 56*, 16 (2018), 5345–5364.

[13] Chen, D., Qi, Q., Li, X., Jin, Z., and Wang, W. A method for studying the influence of different berth layout on port service level based on simulation. In *2021 6th International Conference on Transportation Information and Safety (ICTIS)* (2021), pp. 1315–1319.

[14] Chen, G., and Yang, z. Methods for estimating vehicle queues at a marine terminal: A computational comparison. *International Journal of Applied Mathematics and Computer Science 24* (09 2014), 611–619.

[15] Cheong, C., Tan, K., Liu, D., and Lin, C. Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research 180* (11 2010), 63–103.

[16] Cichowicz, T., Drozdowski, M., Frankiewicz, M., Pawlak, G., Rytwinski, F., and Wasilewski, J. Hyper-heuristics for cross-domain search. *Bulletin of the Polish Academy of Sciences: Technical Sciences 60* (12 2012).

[17] Coffman, E. G., and Garey, M. R. Proof of the 4/3 conjecture for preemptive vs. nonpreemptive two-processor scheduling. *J. ACM 40*, 5 (nov 1993), 991–1018.

[18] Cordeau, J.-F., Laporte, G., Legato, P., and Moccia, L. Models and tabu search heuristics for the berth-allocation problem. *Transportation Science 39* (11 2005), 526–538.

[19] Cruz, R., Mendes, A. B., and Bahiense, L. Schedule robustness in the periodic supply vessels planning problem with stochastic demand and travel time. *International Transactions in Operational Research 31*, 4 (2024), 2338–2365.

[20] Delignette-Muller, M. L., and Dutang, C. fitdistrplus: An R package for fitting distributions. *Journal of Statistical Software 64*, 4 (2015), 1–34.

[21] Dragovic, B., Park, N. K., and Radmilovic, Z. Ship-berth link performance evaluation: Simulation and analytical approaches. *Maritime Policy & Management 33* (07 2006), 281–299.

[22] Dulebenets, M., Kavoosi, M., Abioye, O., and Pasha, J. A self-adaptive evolutionary algorithm for the berth scheduling problem: Towards efficient parameter control. *Algorithms 11* (07 2018), 100.

[23] Ewald, R. *Automatic Algorithm Selection for Complex Simulation Problems*. Fakultät für Informatik und Elektrotechnik, Universität Rostock, 01 2011.

[24] Feitelson, D. G., Tsafrir, D., and Krakov, D. Experience with using the parallel workloads archive. *Journal of Parallel and Distributed Computing 74*, 10 (2014), 2967–2982.

[25] Garey, M. R., and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

[26] Gharehgozli, A., Zaerpour, N., and Koster, R. Container terminal layout design: transition and future. *Maritime Economics & Logistics 22*, 4 (December 2020), 610–639.

[27] Giallombardo, G., Moccia, L., Salani, M., and Vacca, I. Modeling and solving the Tactical Berth Allocation Problem. *Transportation Research Part B: Methodological 44*, 2 (February 2010), 232–245.

[28] Giallombardo, G., Moccia, L., Salani, M., and Vacca, I. Modeling and solving the tactical berth allocation problem. *Transportation Research Part B: Methodological 44* (02 2010), 232–245.

[29] GITHUB. Launch failed requeued held #2. https://github.com/SCOREC/fep/issues/2, 2019. Accessed: 22 Apr. 2024.

[30] GOKKUŞ, U., AND YILDIRIM, M. S. Determination of the optimum berth number of the izmir seaport by deterministic and probabilistic methods based on artificial neural networks. *International Journal of Hybrid Information Technology 8* (09 2015), 11–24.

[31] GOSASANG, V., CHANDRAPRAKAIKUL, W., AND KIATTISIN, S. A comparison of traditional and neural networks forecasting techniques for container throughput at bangkok port. *The Asian Journal of Shipping and Logistics 27* (12 2011), 463–482.

[32] GROUP, N. R. Vehicle routing problem. https://neo.lcc.uma.es/vrp/, 2013. Accessed: 6 Aug. 2024.

[33] HAMMOUTI, I., LAJJAM, A., AND EL MEROUANI, M. Comparison of planning models for dynamic berth allocation problem using a sailfish-based algorithm. *Procedia Computer Science 176* (01 2020), 3112–3120.

[34] HANSEN, P., AND OGUZ, C. A note on formulations of static and dynamic berth allocation problems. *Les Cahiers du GERAD 30* (01 2003), 1–17.

[35] HANSEN, P., OGUZ, C., AND MLADENOVIC, N. Variable neighborhood search for minimum cost berth allocation. *European Journal of Operational Research 191* (12 2008), 636–649.

[36] HEDJAR, R., AND MESSAOUD, B. An automatic collision avoidance algorithm for multiple marine surface vehicles. *International Journal of Applied Mathematics and Computer Science 29* (12 2019), 759–768.

[37] HENDRIKS, M., ARMBRUSTER, D., LAUMANNS, M., LEFEBER, E., AND UDDING, J. Strategic allocation of cyclically calling vessels for multi-terminal container operators. *Flexible Services and Manufacturing Journal - FLEX SERV MANUF J 24* (09 2011), 1–26.

[38] HENDRIKS, M., LAUMANNS, M., LEFEBER, E., AND UDDING, J. Robust cyclic berth planning of container vessels. *Operations Research-Spektrum 32* (07 2010), 501–517.

[39] HOOS, H., KAMINSKI, R., LINDAUER, M., AND SCHAUB, T. aspeed: Solver scheduling via answer set programming. *Theory and Practice of Logic Programming 15*, 1 (2015), 117–142.

[40] HUANG, W.-C., AND WU, S.-C. The estimation of the initial number of berths in a port system based on cost function. *Journal of Marine Science and Technology 13* (03 2005).

[41] IMAI, A., YAMAKAWA, Y., AND HUANG, K. The strategic berth template problem. *Transportation Research Part E: Logistics and Transportation Review 72* (2014), 77–100.

[42] IRIS, K., AND LAM, J. S. L. Optimal energy management and operations planning in seaports with smart grid while harnessing renewable energy under uncertainty. *Omega 103* (03 2021), 102445.

[43] JIANG, X., CHEW, E. P., AND LEE, L. H. *Innovative Container Terminals to Improve Global Container Transport Chains*. Springer, Cham, 12 2015, pp. 3–41.

[44] KADIOGLU, S., MALITSKY, Y., SABHARWAL, A., SAMULOWITZ, H., AND SELLMANN, M. Algorithm selection and scheduling. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming* (01 2011), pp. 454–469.

[45] KANG, L., MENG, Q., AND TAN, K. C. Tugboat scheduling under ship arrival and tugging process time uncertainty. *Transportation Research Part E: Logistics and Transportation Review 144* (2020), 102125.

[46] KASTNER ET AL.(2022)KASTNER, KÄMMERLING, J., AND CLAUSEN. A method for studying the influence of different berth layout on port service level based on simulation. In *Data Science in Maritime and City Logistics: Data-driven Solutions for Logistics and Sustainability. Proceedings of the Hamburg International Conference of Logistics (HICL)* (2022), pp. 485–519.

[47] KHO, F. Kalmar's proposal received a commendation award in the next generation container port challenge. https://www.kalmarglobal.com/news--insights/press_releases/2013/kalmars-proposal-received-a-commendation-award-in-the-next-generation-container-port-challenge-/, 2013. Accessed: 6 Aug. 2024.

[48] Kim, J., and Morrison, J. Offshore port service concepts: Classification and economic feasibility. *Flexible Services and Manufacturing Journal - FLEX SERV MANUF J 24* (09 2012), 1–32.

[49] Kotthoff, L. Algorithm selection for combinatorial search problems: A survey. *AI Magazine 35*, 3 (Sep. 2014), 48–60.

[50] Kovač, N. Metaheuristic approaches for the berth allocation problem. *Yugoslav Journal of Operations Research 27* (01 2017), 1–1.

[51] Lalla-Ruiz, E., Melián-Batista, B., and Marcos Moreno-Vega, J. Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem. *Engineering Applications of Artificial Intelligence 25*, 6 (2012), 1132–1141.

[52] Lasdon, L. S., Fox, R. L., and Ratner, M. W. Nonlinear optimization using the generalized reduced gradient method. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle 8*, V3 (1974), 73–103.

[53] Lee, D.-H., Chen, J. H., and Cao, J. X. The continuous berth allocation problem: A greedy randomized adaptive search solution. *Transportation Research Part E: Logistics and Transportation Review 46* (11 2010), 1017–1029.

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

[55] Li, C., Qi, X., and Song, D. Real-time schedule recovery in liner shipping service with regular uncertainties and disruption events. *Transportation Research Part B: Methodological 93* (2016), 762–788. Maritime Logistics.

[56] Li, M.-W., Hong, W.-C., Geng, J., and Wang, J. Berth and quay crane coordinated scheduling using multi-objective chaos cloud particle swarm optimization algorithm. *Neural Comput. Appl. 28*, 11 (nov 2017), 3163–3182.

[57] Lim, A. The berth planning problem. *Operations Research Letters 22*, 2 (1998), 105–110.

[58] Liu, B., Li, Z.-C., Sheng, D., and Wang, Y. Integrated planning of berth allocation and vessel sequencing in a seaport with one-way navigation channel. *Transportation Research Part B Methodological 143* (01 2021).

[59] Liu, C. Iterative heuristic for simultaneous allocations of berths, quay cranes, and yards under practical situations. *Transportation Research Part E: Logistics and Transportation Review 133* (2020), 101814.

[60] Maciej, D., Jakub, W., and Eric, S. Resources for algorithm selection for large berth allocation problem. http://www.cs.put.poznan.pl/mdrozdowski/asp-bap/, 2020. Accessed: 6 Aug. 2024.

[61] Martin Alcalde, E., Kim, K. H., and Marchán, S. S. Optimal space for storage yard considering yard inventory forecasts and terminal performance. *Transportation Research Part E: Logistics and Transportation Review 82* (2015), 101–128.

[62] Moorthy, R., and Teo, C. Berth management in container terminal: The template design problem. *OR Spectrum 28* (01 2006), 495–518.

[63] Nam, K., Kwak, K., and Yu, M. Simulation study of container terminal performance. *Journal of Waterway Port Coastal and Ocean Engineering 128* (05 2002), 126–132.

[64] O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., and O'Sullivan, B. Using case-based reasoning in an algorithm portfolio for constraint solving? *Irish Conference on Artificial Intelligence and Cognitive Science* (03 2013).

[65] Pachakis, D., and Kiremidjian, A. Ship traffic modeling methodology for ports. *Journal of Waterway Port Coastal and Ocean Engineering-asce - J WATERW PORT COAST OC-ASCE 129* (09 2003).

[66] Petering, M. Effect of block width and storage yard layout on marine container terminal performance. *Transportation Research Part E: Logistics and Transportation Review 45* (07 2009), 591–610.

[67] Piechowiak, K., Drozdowski, M., and Éric Sanlaville. Framework of algorithm portfolios for strip packing problem. *Computers & Industrial Engineering 172* (2022), 108538.

[68] Reid, M. Reliability is a python library for reliability engineering, revision c4af6a17. https://reliability.readthedocs.io/en/latest/index.html, 2023. Accessed: 9 Apr. 2024,.

[69] Reinelt, G. Tsplib. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html, 1995. Accessed: 6 Aug. 2024.

[70] Rice, J. The algorithm selection problem. Tech. Rep. 99, Purdue University, Computer Science Technical Reports, 1975.

[71] Rodrigues, F., and Agra, A. Berth allocation and quay crane assignment/scheduling problem under uncertainty: A survey. *European Journal of Operational Research 303*, 2 (2022), 501–524.

[72] Rodrigues, F., and Agra, A. Handling uncertainty in the quay crane scheduling problem: a unified distributionally robust decision model. *International Transactions in Operational Research 31*, 2 (2024), 721–748.

[73] Rodríguez-Molins, M., Salido, M., and Barber, F. A grasp-based metaheuristic for the berth allocation problem and the quay crane assignment problem by managing vessel cargo holds. *Applied Intelligence 40* (03 2014), 273–290.

[74] Schepler, X., Absi, N., Feillet, D., and Sanlaville, E. The stochastic discrete berth allocation problem. *EURO Journal on Transportation and Logistics 8* (07 2018).

[75] Schepler, X., Balev, S., Michel, S., and Éric Sanlaville. Global planning in a multi-terminal and multi-modal maritime container port. *Transportation Research Part E: Logistics and Transportation Review 100* (2017), 38–62.

[76] Shabayek, A., and Yeung, W. A simulation model for the kwai chung container terminals in hong kong. *European Journal of Operational Research 140* (02 2002), 1–11.

[77] Slurm Support. Bug 3535 - jobs stuck in "launch failed requeued held". https://bugs.schedmd.com/show_bug.cgi?id=3535, 2017. Accessed: 22 Apr. 2024.

[78] Smith-Miles, K., and Lopes, L. Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research 39*, 5 (2012), 875–889.

[79] Smith-Miles, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys 41*, 1 (jan 2009).

[80] Souravlias, D., Parsopoulos, K., Kotsireas, I., and Pardalos, P. *Algorithm Portfolios: Advances, Applications, and Challenges*. Springer Cham, 12 2020.

[81] Stahlbock, R., and Voss, S. Operations research at container terminals: a literature update. *OR Spectrum 30* (2008), 1–52.

[82] Taillard, E. Benchmarks for basic scheduling problems. *European Journal of Operational Research 64*, 2 (1993), 278–285. Project Management anf Scheduling.

[83] Tan, Z., Zhang, Q., Yuan, Y., and Jin, Y. A decision method on yard cranes transformation and deployment in green ports. *International Transactions in Operational Research 29* (06 2021).

[84] Ting, C.-J., wu, k.-c., and Chou, H. Particle swarm optimization algorithm for the berth allocation problem. *Expert Systems with Applications: An International Journal 41* (03 2014), 1543–1550.

[85] Top 500. Altair - ch121l v5 liquid-cooled, xeon platinum 8268 24c 2.9GHz,. https://top500.org/system/179903/, 2023. Accessed: 25 Apr. 2024.

[86] Vernimmen, B., Dullaert, W., and Engelen, S. Schedule unreliability in liner shipping: Origins and consequences for the hinterland supply chain. *Maritime Economics and Logistics 9* (09 2007), 193–213.

[87] Wang, W., Chen, X., Musial, J., and Blazewicz, J. Two meta–heuristic algorithms for scheduling on unrelated machines with the late work criterion. *International Journal of Applied Mathematics and Computer Science 30* (09 2020), 573–584.

[88] Wawrzyniak, J., Drozdowski, M., and Sanlaville, É. Selecting algorithms for large berth allocation problems. *European Journal of Operational Research 283*, 3 (2020), 844–862.

[89] Wawrzyniak, J., Drozdowski, M., and Sanlaville, É. Container ship traffic model for simulation studies - additional resources, 2021. Accessed: 25 Apr. 2024.

[90] Wawrzyniak, J., Drozdowski, M., and Sanlaville, E. Container ship traffic model for simulation studies. *International Journal of Applied Mathematics and Computer Science 32*, 4 (2022), 537–552.

[91] Wawrzyniak, J., Drozdowski, M., Sanlaville, E., Pigné, Y., and Guinand, F. Quay partitioning problem. *International Transactions in Operational Research 31*, 3 (2024), 1554–1584.

[92] Wawrzyniak, J., Drozdowski, M., and Éric Sanlaville. Heuristics for long time horizon berth allocation problem. Tech. rep., Institute of Computing Science, Poznań University of Technology, 2017.

[93] Wawrzyniak, J., Drozdowski, M., and Éric Sanlaville. A container ship traffic model for simulation studies. *International Journal of Applied Mathematics and Computer Science 32*, 4 (2022), 537–552.

[94] Wawrzyniak et al.(2022b)Wawrzyniak, D., and Sanlaville. Quay partitioning problem - additional resources. http://www.cs.put.poznan.pl/mdrozdowski/qpp/, 2022. Accessed: 6 Aug. 2024.

[95] Wiese, J., and Kliewer, N. Planning container terminal layouts considering equipment types and storage block design. *Operations Research/ Computer Science Interfaces Series 49* (02 2011).

[96] Wikipedia contributors. Port of Singapore. https://en.wikipedia.org/wiki/Port_of_Singapore, 2022. Accessed: 6 Aug. 2024.

[97] Wikipedia contributors. Automatic identification system. https://en.wikipedia.org/wiki/Automatic_identification_system, 2023. Accessed: 6 Aug. 2024.

[98] Wikipedia contributors. Big O notation. https://en.wikipedia.org/wiki/Big_O_notation, 2023. Accessed: 6 Aug. 2024.

[99] Wikipedia contributors. Yangshan port. https://en.wikipedia.org/wiki/Yangshan_Port, 2023. Accessed: 6 Aug. 2024.

[100] Wikipedia contributors. Bayesian information criterion. https://en.wikipedia.org/wiki/Bayesian_information_criterion, 2024. Accessed: 9 Apr. 2024.

[101] Wikipedia contributors. Censoring (statistics). https://en.wikipedia.org/wiki/Censoring_(statistics), 2024. Accessed: 9 Apr. 2024.

[102] Xiang, X., and Changchun, L. An almost robust optimization model for integrated berth allocation and quay crane assignment problem. *Omega 104* (03 2021), 102455.

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

[104] Zhen, L., Chew, E. P., and Lee, L. H. An integrated model for berth template and yard template planning in transshipment hubs. *Transportation Science 45* (11 2011), 483–504.

[105] Zhen, L., Liang, Z., Zhuge, D., Lee, L. H., and Chew, E. P. Daily berth planning in a tidal port with channel flow control. *Transportation Research Part B: Methodological 106* (2017), 193–217.

[106] Zhou, Y., Wang, W., Song, X., and Guo, Z. Simulation-based optimization for yard design at mega container terminal under uncertainty. *Mathematical Problems in Engineering 2016* (09 2016).