Solving Berth Allocation Problems via Selective Graph Colouring

Alberto Santini[1]

[1]Department of Economics and Business, Universitat Pompeu Fabra, Barcelona, 08005 Spain

November 20, 2025

Abstract

Berth Allocation Problems (BAPs) are a family of well-known problems in port operations. They involve assigning incoming container ships to port berths. In the main version we consider, simply denoted BAP, the objective is to minimise the makespan of loading and unloading operations. The decision version of the BAP (DBAP) asks if it is possible to load and unload a set of ships within a specified deadline. We solve the BAP by progressively refining upper and lower bounds of the optimal makespan and use the DBAP to determine whether a feasible solution with a given maximum makespan exists. This method relies on being able to solve multiple DBAPs quickly. To this end, we use two techniques: we transform the DBAP into the Selective Graph Colouring Problem (SGCP) and solve it using a state-of-the-art branch-and-price algorithm, and we perform a dynamic discretisation of the time horizon that reduces the size of the solved SGCP instances. Compared with five formulations from the literature, the proposed exact approach finds more optimal solutions within a one-hour time limit.

Keywords: Berth allocation problem, selective graph colouring, branch-and-price, maritime logistics.

1 Introduction

Berth Allocation Problems (BAPs) are a family of operational problems arising in port container terminal management. In a popular version, the port quay is divided into discrete berths, and incoming container ships must dock at available berths to load and unload their cargo. Each ship can require one or more berths for a prespecified amount of time, depending on its size. If no suitable berth is available when a ship reaches the port, it must wait. Figure 1 shows a quay divided into five berths and a moored ship occupying two. Port managers must decide when and where each ship can berth to minimise a given objective function, such as the total makespan (i.e., the completion time of the last served ship) or the sum of the waiting times.

Different BAPs make different assumptions about how the port operates and the data available to the decision-maker. Bierwirth and Meisel (2015) proposes a classification along four attributes: spatial, temporal, handling time, and objective function.

Regarding the spatial attribute, problems belong to one of three classes. In the first, the quay is continuous and ships can berth at any point. In the second, it is divided into berths, and each ship occupies a single berth. In the third, the quay is divided into berths, and depending on its size, a ship can occupy more than one berth, or multiple ships can occupy a single berth. Our work mainly considers the third class ("hybr" in the classification). While our approach can also solve problems falling in the second class ("disc") as a special case, it does not explicitly exploit the limitation that each ship can only occupy one berth.

Concerning the temporal attribute, the two main classes of deterministic problems are: static, if all ships are assumed to be ready to berth at the beginning of the time horizon and dynamic, if ships have individual arrival times. We consider the dynamic case ("dyn" in the classification), although our approach also works in the static case ("stat").

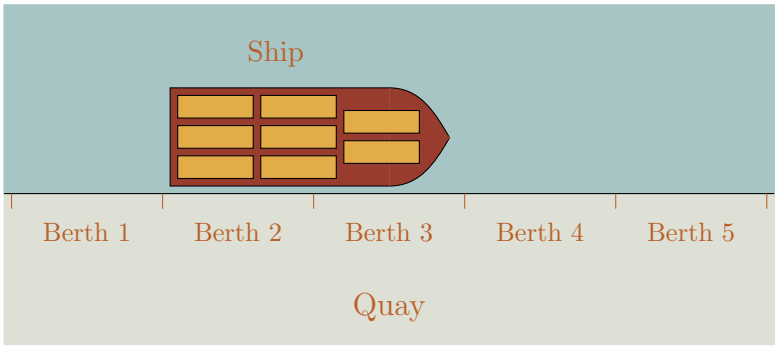

Figure 1: A quay divided into five berths, and a container ship occupying two of them.

The third attribute is the handling time. It can either depend on the ship ("fix" in the classification), on the ship and the assigned berth ("pos" in the classification), or on more complex decisions such as the number of assigned cranes (see, e.g., Vacca et al. 2012; Santini, Friberg, et al. 2015; Cahyono et al. 2022; Y. Li et al. 2022; Ji et al. 2023). Our proposed approach can solve problems in the "fix" and "pos" classes.

The fourth attribute defines the objective function to minimise. Our approach can be used to minimise either the makespan ("max(comp)" in the classification) or the total quay length required to handle all ships within a given deadline ("max(pos)"). Other popular objective functions include "$\sum$(wait+hand)" that minimises the total waiting and handling times and "$\sum$(wait+hand+tard)" that also includes tardiness.

In summary, according to the definition of Bierwirth and Meisel (2015), our proposed approach is best suited for problems of type "hyb|stat,dyn|fix,pos|max(compl),max(pos)", in which comma-separated classes indicate that the approach is valid for all the classes of a given attribute. The objective function "max(pos)" was introduced by Lim (1988) but is not popular in recent literature. Objective function "max(compl)" has received more attention (see, e.g., Lee and Wang 2010; Błażewicz et al. 2011; Emde et al. 2014). Indeed, in our computational experiments and due to data availability, we focus on the "hyb|dyn|fix|max(compl)" problem type.

Figure 2 places our proposed approach in the existing literature on BAP variants. We present the variants hierarchically, distinguishing them by their objective function, and then by their spatial, temporal and handling attributes. Although our approach can solve several variants, we focus on the max(compl) objective function rather than max(pos), because of its greater popularity and real-world relevance. We also focus on the hybrid spatial attribute because the resulting problem is more general than the discrete case and more realistically represents modern container terminals compared to the continuous case. We consider the dynamic temporal attribute because it generalises the static one. Finally, although our approach can solve problems with position-based handling times, we could not find any real-world data involving this attribute. Therefore, we use instances with fixed handling times in our computational experiments.

After a comprehensive literature analysis, we identified five exact approaches which, despite having been devised for different problem variants, can be adapted to the problem we consider. These are: the relative position (RP) and position assignment (PA) formulations of Guan and Cheung (2004); the generalised set partitioning problem (GSPP) formulation of Buhrkal et al. (2011); the sequence variables (S) and time index (TI) models proposed by Ernst et al. (2017). We discuss the existing literature further in Section 2.

Formal definition. We are given a set of n ships and a time horizon discretised into T intervals. Each ship i has an associated length $\ell_i > 0$, arrival time $a_i \in \{1, \dots T\}$, and handling time $h_i > 0$. The quay of length L is divided into m adjacent berths. Each berth j has an associated length $\lambda_j > 0$ and starts at a distance of $\delta_j \geq 0$ from the leftmost point of the quay ($\delta_1 = 0$, $\delta_j = \sum_{k=1}^{j-1} \lambda_k$). Ship i can moor with its leftmost end at berth j at time t if $\delta_j + \ell_i \leq L$ (the ship fits in the quay), $t \geq a_i$ (the ship

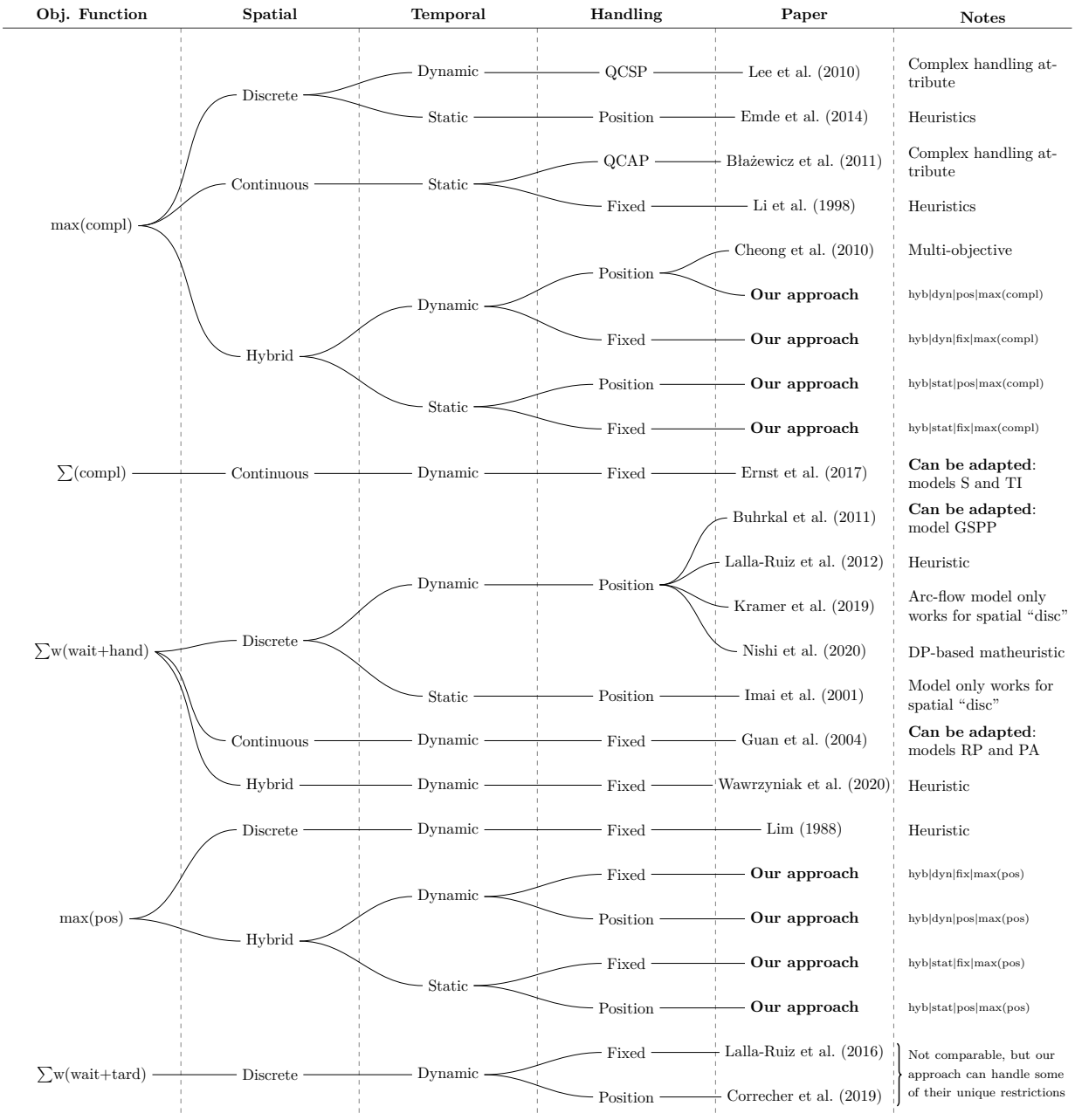

Figure 2: Positioning of our proposed methodology in the family of BAPs, with links to the relevant literature.

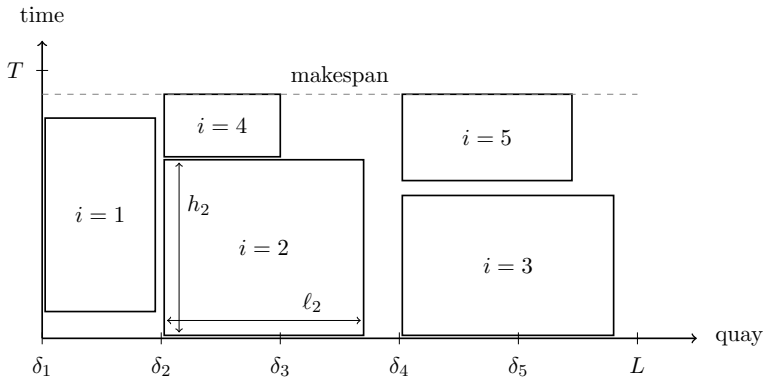

Figure 3: Geometric representation of five ships with feasible assignments.

berths not earlier than its arrival), $t + h_i \leq T$ (the cargo can be handled within the end of the time horizon), and the berths occupied by the ship are free of other ships during the entire handling period. We can represent the quay on the horizontal axis and the time on the vertical axis of a Cartesian plane, such as in Figure 3. Mooring ship i at berth j at time t corresponds to drawing a rectangle with the bottom left corner at coordinates (δ_j, t) and the top right corner at coordinates $(\delta_j + \ell_i, t + h_i)$. The condition that no two ships occupy the same berth simultaneously implies that no two rectangles overlap. The vertical coordinate of the upper side of the topmost rectangle gives the makespan.

The decision version of the BAP. The BAP asks to find a feasible assignment of ships to berths (called a *berthing plan*) that minimises the makespan. Given a deadline $M \in \{1, \ldots, T\}$, the decision version of the BAP (DBAP) asks if there is a berthing plan such that the corresponding makespan does not exceed M. Graphically, it asks if it is possible to pack without overlap n rectangles, each of size $\ell_i \times h_i$, into another rectangle of size $L \times M$ without violating any of the BAP constraints. We denote with $\text{DBAP}(M) \in \{\text{True}, \text{False}\}$ the solution of the DBAP with deadline M.

Key contribution. This work's key contribution is introducing an exact BAP solution method that relies on solving multiple DBAPs. For hard instances, this method outperforms the RP, PA, S, TI, and GSPP formulations by solving more instances. Our method relies on the observation that z^* is the optimal solution of the BAP if $\text{DBAP}(z^*) = \text{True}$ and $\text{DBAP}(z^* - 1) = \text{False}$. We propose to find the value z^* by devising increasingly tight lower and upper bounds on its value. We iteratively tighten such bounds with a tailored search algorithm in the interval $\{H, \ldots, T\}$, where $H = \max_{i \in \{1,\ldots,n\}}(a_i + h_i)$ is the earliest possible makespan if all ships berth immediately upon arrival. For the success of this algorithm, it is crucial to devise a fast solution method for the DBAP. In this work, we propose transforming the DBAP instances into Selective Graph Colouring Problem (SGCP) instances and solving them using a state-of-the-art exact algorithm. Furthermore, we reduce the size of the solved SGCP instances by performing a dynamic time discretisation that starts with a coarser version of the time horizon $\{1, \ldots, T\}$ and progressively returns to the original discretisation when the bounds approach the optimum z^*.

Paper organisation. The rest of the paper is organised as follows. Section 2 positions this paper in the current literature on BAPs. In Section 3, we introduce the SGCP and show how it can solve the DBAP. We also briefly describe the branch-and-price algorithm for the SGCP devised by Furini et al. (2018). We introduce the bound-based search algorithm and the dynamic time discretisation method in Section 4. Section 5 describes computational results on a diverse set of instances based on real-life data from the port of Barcelona. Finally, Section 6 draws some conclusions and outlines future research directions.

2 Previous works

Most problems in the BAP family are $\mathcal{NP}$-hard (Hansen and Oğuz 2003), and they have proven challenging to solve exactly. For example, the tree search method of Guan and Cheung (2004) takes over 15 hours (on average) for instances with 10 berths and 15 ships when minimising the weighted flow time. The compact formulation of Hansen and Oğuz (2003) takes over 7 hours (on average) for instances with five berths and 30 ships, minimising the sum of waiting and handling times. The Lagrangian algorithm introduced by Imai et al. (2001) is faster (in the order of tens or hundreds of seconds) but has average gaps of 51% already for instances with five berths and 25 ships, again minimising the sum of waiting and handling times.

In our literature analysis, we found six formulations for problems in the BAP family which are competitive for at least some classes of instances. Out of them, five can be adapted to the main problem type we consider, "hyb|dyn|fix|max(compl)", whereas the arc-flow model of Kramer et al. (2019) strictly relies on the discrete space attribute. In the following, we provide a short résumé of these six formulations.

Guan and Cheung (2004) introduced two formulations for a BAP of type "hyb|dyn|fix|$\sum$w(wait+hand)" in which all berths have the same length. The first formulation, called Relative Position (RP), uses binary variables to keep track of pairs of ships such that the first completes before the second moors (along the temporal dimension), and such that the rightmost end of the first ship is on the left of the leftmost end of the second (along the spatial dimension). The second formulation, called Position Assignment (PA), is a hybrid packing-covering formulation with binary variables keeping track of the bottom-left corner of each ship's rectangle. We describe these formulations, respectively, in Appendix A.1 and Appendix A.2. In their work, the authors use RP as the main formulation and explore its solution space via tree search. A Lagrangian relaxation of PA is used during the tree's exploration to provide bounds and prune parts of the tree. The authors also include improvement heuristics to enhance the tree search performance. They test their approach on randomly generated instances with 10 berths, 168 intervals (one week, discretised hour-by-hour), and up to 80 ships.

Buhrkal et al. (2011) describes a formulation first introduced by Christensen and Holst (2008) in their M.Sc. thesis. This is a generalised set partitioning problem (GSPP) formulation with one variable for each possible ship, berth, and feasible mooring time of the ship at the berth. We present it in Appendix A.3. Despite the large number of columns, the GSPP formulation enjoys an especially tight continuous relaxation that makes it extremely competitive, outperforming the previous state of the art (Monaco and Samarra 2007). However, it has two downsides. First, most practically relevant side-constraints involving more than one ship are hard (or outright impossible) to include in a GSPP-based formulation. By contrast, constraints involving single vessels can be incorporated straightforwardly, as shown, e.g., by Lalla-Ruiz, Expósito-Izquierdo, et al. (2016). Second, despite the number of variables being polynomial in the input data (number of ships, number of berths, time horizon length), it tends to be very large. Indeed, in a subsequent study (Kramer et al. 2019), it was shown that building the GSPP model resulted in out-of-memory errors for instances with 200–250 ships on a machine with 16GB of RAM.

Ernst et al. (2017) originally considered a "cont|dyn|fix|$\sum$(compl)" BAP characterised by tidal constraints, i.e., a set of time windows during which the ships cannot leave the port. In Appendix A.4 and Appendix A.5, we describe variants of the two formulations presented by the authors, adapting them to the case of a discrete berth. The first is the sequence variables formulation (S), which uses two main sets of binary variables defined for each pair of ships. One set of variables identifies pairs of ships such that the first one completes before the second one starts processing, while the other set keeps track of ships with overlapping handling periods. The second is the time index formulation (TI), which takes this name because it includes one set of binary variables indexed over the ships and the time intervals, taking the value one if and only if the given ship completes handling by the given interval. Both formulations are strengthened with a large number of valid inequalities. The TI formulation outperforms S on randomly-generated instances incorporating tide times from the port of Newcastle, Australia. These instances include up to 32 ships and 168 or 336 time intervals (one or two weeks,

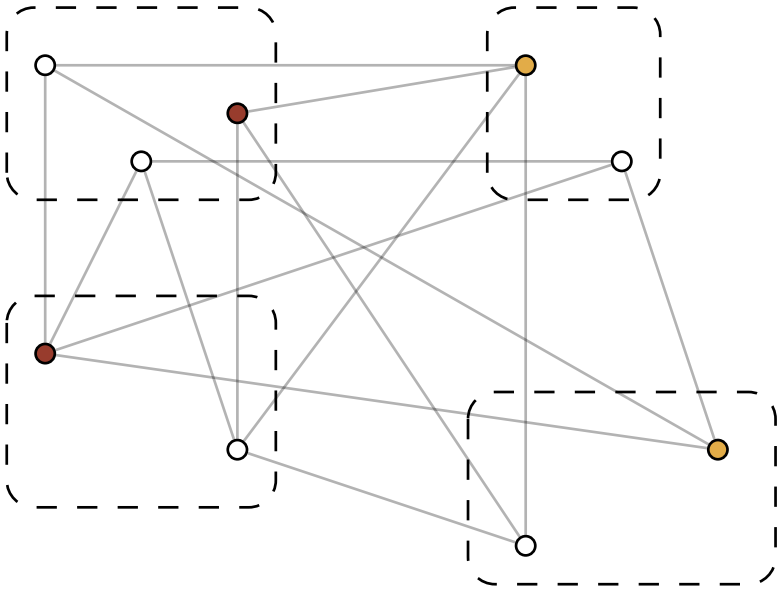

Figure 4: Example SGCP instance. Dashed rectangles represent partitions, and solid grey lines are edges. A possible optimal solution using brown and yellow colours is depicted.

discretised hour-by-hour).

Kramer et al. (2019) introduced an arc-flow formulation for a "disc|dyn|pos|$\sum$w(wait+hand)" BAP. In their approach, each berth is a resource flowing through time; its path defines the ships served by the given berth. Because of this interpretation, the formulation heavily relies on the "disc" spatial attribute, i.e., that each ship occupies exactly one berth. On instances of this type, the arc-flow formulation is very competitive, even outperforming the previous state-of-the-art GSPP formulation.

Finally, as a consequence of the difficulty of solving the BAP exactly, we remark that most recent research has focused on heuristic methods, such as the first-fit-decreasing heuristic of C.-L. Li et al. (1998), the genetic algorithm of Lee and Chen (2010) and the rounding heuristic of Błażewicz et al. (2011) for BAPs with handling times based on quay-crane assignment, the evolutionary algorithm of Cheong et al. (2010), the hybrid heuristic of Lalla-Ruiz, Melián-Batista, et al. (2012), the iterated local search of Correcher et al. (2019), the dynamic-programming-based matheuristic of Nishi et al. (2020), and the algorithm portfolio approach of Wawrzyniak et al. (2020).

3 Selective graph colouring and berth allocation

The SGCP, also known as the *Partition Colouring Problem* (Hoshino et al. 2011; Cornaz et al. 2019; Zhu et al. 2022), is a combinatorial optimisation problem defined on an undirected graph $G = (V, E)$ with vertices V and edges $E \subseteq V^2$, where the vertex set has an associated partition $\mathcal{P}$, i.e., $\bigcup \mathcal{P} = V$ and $P \cap Q = \emptyset \; \forall P, Q \in \mathcal{P}$, $P \neq Q$. The SGCP consists of selecting exactly one vertex in each partition set and assigning it a colour. The objective of the problem is to minimise the number of colours used, subject to the requirement that adjacent vertices cannot have the same colour. As a generalisation of the classical Vertex Colouring Problem, the SGCP is $\mathcal{NP}$-hard.

Figure 4 shows a small SGCP instance with nine vertices (represented by circles) partitioned into four sets (dashed rectangles). Solid grey lines denote edges. The figure shows an optimal solution using two colours (brown and yellow).

In the rest of this section, we explain how, for a given DBAP instance, we create a corresponding SGCP instance with the property that the DBAP solution is positive if and only if the optimal solution of the SGCP uses one colour. Finally, we briefly outline the exact algorithm developed by Furini et al. (2018) to solve the SGCP.

We first build the vertex set as

$$V^{\mathrm{BAP}} = \{(i,j,t) \ : 1 \le i \le n,\ 1 \le j \le m,\ a_i \le t \le T \text{ and } \\ \delta_j + \ell_i \le L,\ t + h_i \le T\}.$$

Vertex (i,j,t) corresponds to ship i mooring at berth j at time t. We partition V^{BAP} by grouping vertices corresponding to the same ship, i.e., $\mathcal{P} = \{V_1, \ldots, V_n\}$ where $V_i = \{(i',j',t') \in V^{\mathrm{BAP}} \ : \ i' = i\}$. To build the edge set E^{BAP}, we add an edge between vertices (i,j,t) and (i',j',t'), for $i \neq i'$, if the corresponding rectangles $[t, t+h_i] \times [\delta_j, \delta_j + \ell_i]$ and $[t', t'+h_{i'}] \times [\delta_{j'}, \delta_{j'} + \ell_{i'}]$ overlap. Such an edge stands for an incompatible assignment that would cause the same berth to be occupied by two ships simultaneously. Because each ship must be assigned a single mooring berth (i.e., the berth where the leftmost end of the ship starts) and a single mooring time, we must select exactly one vertex in each partition set. If the optimal solution of the SGCP uses one colour, it is possible to select a mooring berth and a mooring time for each ship so that all ships can be accommodated within the end of the time horizon. In this case, the solution of DBAP is True. Conversely, if the optimal solution of the SGCP uses two or more colours, the solution of DBAP is False.

Demange et al. (2015) first highlighted the relationship between the BAP and the SGCP in a survey on SGCP applications. The authors remarked that "It would be interesting to explore graph theoretical approaches [...] to solve the berth allocation problem". This work aims at filling this gap, providing a computational study to assess the potential of using the SGCP to solve the BAP and completing the preliminary work sketched in the PhD thesis of the author (Santini 2017). In (Demange et al. 2015), the authors also remark that the SGCP remains $\mathcal{NP}$-hard when restricted to graphs $(V^{\mathrm{BAP}}, E^{\mathrm{BAP}})$ because these graphs generalise interval graphs, where the SGCP is $\mathcal{NP}$-complete. We highlight that the correspondence between DBAP and the SGCP allows modelling additional berthing constraints. For example, we can model ship–berth draft compatibilities (Grubišić and Barić 2015; Liu et al. 2018; Malaguti et al. 2018) by removing vertices corresponding to incompatible ship–berth pairs.

3.1 Solving a DBAP instance through a transformation to the SGCP

In our approach, we solve the SGCP instance associated with the corresponding DBAP instance using the branch-and-price algorithm of Furini et al. (2018). If, during the exploration of the branch-and-bound tree, the algorithm finds a feasible solution using one colour, we terminate and report that DBAP's solution is True. Similarly, if at any moment the best dual bound is strictly greater than one, we terminate the algorithm and report that DBAP's solution is False.

The algorithm relies on an extended formulation for the SGCP, which we summarise here for a generic undirected graph $G = (V, E)$. Let $\mathcal{S}$ be the collection of stable sets of G that intersect each partition set in at most one vertex, i.e.,

$$\mathcal{S} = \big\{S \subseteq V \ : \{v_1, v_2\} \notin E, \forall v_1, v_2 \in S \text{ and } \\ |S \cap P| \le 1, \forall P \in \mathcal{P}\big\}.$$

The formulation then uses binary variables x_S for $S \in \mathcal{S}$, taking value 1 if and only if all vertices of S are coloured with the same colour, and reads as follows:

$$\min \quad \sum_{S \in \mathcal{S}} x_S \tag{1}$$

$$\text{s.t.} \quad \sum_{\substack{S \in \mathcal{S} \\ |S \cap P| \ge 1}} x_S = 1 \qquad \forall P \in \mathcal{P} \tag{2}$$

$$x_S \in \{0,1\} \qquad \forall S \in \mathcal{S}. \tag{3}$$

The objective function (1) minimises the number of colours used. Constraint (2) ensures that one vertex in each partition set is coloured.

DBAP($\underline{M}$)	DBAP($\bar{M}$)	New $[\underline{z}, \bar{z}]$	New $[\underline{z}, \bar{z}]$ with tightened bounds
True	True	$[\underline{z}, \underline{M}]$	$[\underline{z}, z_{\text{DBAP}(\underline{M})}]$
False	True	$[\underline{M}, \bar{M}]$	$[\underline{M}, z_{\text{DBAP}(\bar{M})}]$
False	False	$[\bar{M}, \bar{z}]$	$[\bar{M}, \bar{z}]$

Table 1: Update rule for bounds $\underline{z}$ and $\bar{z}$ in the binary search algorithm.

The branch-and-price algorithm starts with a reduced subset of variables and introduces new ones via column generation at each branch-and-bound node. To detect new columns with negative reduced cost, the authors solve a Maximum Weight Stable Set Problem (Held et al. 2012). Carefully chosen branching rules ensure that the subproblem nature does not change during the exploration of the tree. Furthermore, the authors initialise the column pool with stable sets produced by a Tabu Search algorithm (Glover and Laguna 1997) and an Adaptive Large Neighbourhood Search algorithm (Ropke and Pisinger 2006) with the Worse Accept acceptance criterion (Santini, Ropke, et al. 2018).

4 Bound-based binary search algorithm

This section presents an algorithm to find the BAP optimal solution $z^* \in \{H, \dots, T\}$. We first present a basic version in Section 4.1 and then improve it via dynamic time discretisation in Section 4.2.

4.1 Base algorithm

The algorithm we introduce keeps track of two bounds, $\underline{z}$ and $\bar{z}$ such that $\underline{z} < z^* \leq \bar{z}$. When $\underline{z} = \bar{z} - 1$, the optimum of the BAP is $z^* = \bar{z}$ and the algorithm terminates.

Initially, we set $\underline{z} = H - 1$ and $\bar{z} = T$. At each iteration, we consider the two values $\underline{M}$ and $\bar{M}$ at one-third and two-thirds of the interval $[\underline{z}, \bar{z}]$. More precisely, we let

$$\underline{M} = \left\lfloor \frac{1}{3}(\bar{z} - \underline{z}) \right\rfloor \quad \text{and} \quad \bar{M} = \left\lceil \frac{2}{3}(\bar{z} - \underline{z}) \right\rceil.$$

We solve DBAP($\underline{M}$) and DBAP($\bar{M}$) using the SGCP, as explained in Section 3.1. We then update the values of $\underline{z}$ and $\bar{z}$ as outlined in the first three columns of Table 1. The algorithm terminates when the criterion $\underline{z} = \bar{z} - 1$ is met.

Recovering the BAP solution from the SGCP solution. Assume that DBAP(M) = True for some value of M. In this case, the corresponding SGCP instance has a solution using one colour. Let $P = \{(1, j_1, t_1), \dots, (n, j_n, t_n)\}$ be the only independent set making up the SGCP solution. The corresponding BAP berthing plan assigns the berth j_i and the mooring time t_i to each ship $i \in \{1, \dots, n\}$. The makespan of this solution is

$$z_{\text{DBAP}(M)} = \max_{i \in \{1,\dots,n\}} \{t_i + h_i\}.$$

Bound tightening. When DBAP(M) = True, there is a solution to the BAP whose makespan is at most M. However, the makespan $z_{\text{DBAP}(M)}$ of the corresponding berthing plan can be strictly smaller than M when the BAP solution does not use the entire planning horizon. In this case, using the BAP makespan instead of M in the binary search algorithm is convenient because the makespan provides tighter bounds when updating the interval $[\underline{z}, \bar{z}]$. The fourth column of Table 1 provides the details of this update.

4.2 Dynamic time discretisation

A key difficulty in solving DBAP by transformation to the SGCP is that the resulting vertex set V^{BAP} is large; indeed, $|V^{\text{BAP}}| = \mathcal{O}(nmT)$. In practical applications, term T is larger than n and m.

Therefore, in this section, we introduce a time discretisation technique to reduce the number of periods considered and, thus, the size of V^{BAP}.

Given a coefficient $\alpha \in \mathbb{N}^+$, we introduce two methods to transform a DBAP instance into a new one with a more coarse time discretisation, containing roughly T/α periods. We denote the two resulting instances DBAP^1_α and DBAP^2_α. Their solutions, given a deadline M, are denoted $\text{DBAP}^1_\alpha(M)$ and $\text{DBAP}^2_\alpha(M)$.

Properties. The two discretisations satisfy the following properties for any $\alpha \geq 2$. If $\text{DBAP}^1_\alpha(M) = \text{True}$, then $\text{DBAP}(M) = \text{True}$; however, if $\text{DBAP}^1_\alpha(M) = \text{False}$, $\text{DBAP}(M)$ could be either True or False. Conversely, if $\text{DBAP}^2_\alpha(M) = \text{False}$, then $\text{DBAP}(M) = \text{False}$; however, if $\text{DBAP}^2_\alpha(M) = \text{True}$, $\text{DBAP}(M)$ could be either True or False. Intuitively, in DBAP^1_α, the problem data is modified conservatively (i.e., pessimistically) producing a more constrained instance. Therefore, if DBAP^1_α is feasible for a given M, the original DBAP instance is also feasible. Conversely, in DBAP^2_α, the problem data is modified optimistically producing a relaxed instance. Therefore, if DBAP^2_α is infeasible for a given M, the original DBAP instance is also infeasible.

Formal definition. Instance DBAP^1_α is obtained from the following modifications of the original DBAP instance: (i) the number of periods T is replaced by $\lfloor T/\alpha \rfloor$; (ii) each ship arrival time a_i is replaced by $\lceil a_i/\alpha \rceil$; (iii) each ship handling time h_i is replaced by $\lceil h_i/\alpha \rceil$; (iv) the deadline M is replaced by $\lfloor M/\alpha \rfloor$. Conversely, in instance DBAP^2_α, the number of periods is replaced by $\lceil T/\alpha \rceil$, the arrival times are $\lfloor a_i/\alpha \rfloor$, the handling times are $\lfloor h_i/\alpha \rfloor$, and the deadline is $\lceil M/\alpha \rceil$. We remark that, for $\alpha = 1$, the above definition guarantees that both DBAP^1_α and DBAP^2_α correspond to the original instance.

Primal bound from the discretised instance. When $\text{DBAP}^1_\alpha(\bar{M}) = \text{True}$ for a deadline $\bar{M}$, we know that $\text{DBAP}(\bar{M}) = \text{True}$. Therefore, $\bar{M}$ is an upper bound on the optimum z^*. However, in Section 4.1, we have shown that this bound can be improved to $z_{\text{DBAP}(M)}$ when solving $\text{DBAP}(M)$. We can also improve the bound when solving $\text{DBAP}^1_\alpha(\bar{M})$. To this end, let $P^1 = \{(1, j_1, t_1), \ldots, (n, j_n, t_n)\}$ be the only independent set of the graph associated with the solution of $\text{DBAP}^1_\alpha(\bar{M})$. We recall that DBAP^1_α has a coarser time discretisation than DBAP and a period t in solution P^1 corresponds to a period αt in the original discretisation of DBAP. We can transform P^1 into a feasible solution for the original BAP instance, i.e., into $P = \{(1, j_1, \alpha t_1), \ldots, (n, j_n, \alpha t_n)\}$. The makespan of this solution,

$$z_{\text{DBAP}^1_\alpha(M)} = \max_{i \in \{1,\ldots,n\}} \{\alpha t_i + h_i\},$$

is a valid upper bound for z^*.

Usage. When using dynamic time discretisation, we compute $\text{DBAP}^2_{\alpha_2}(\underline{M})$ instead of $\text{DBAP}(\underline{M})$ and $\text{DBAP}^1_{\alpha_1}(\bar{M})$ instead of $\text{DBAP}(\bar{M})$, where α_1 and α_2 are two integer numbers. In our implementation, both parameters initially start at the same value, i.e., $\alpha_1 = \alpha_2 = \alpha^{(0)}$. Our algorithm uses a sequence of Q integer numbers $\alpha^{(0)}, \alpha^{(1)}, \ldots, \alpha^{(Q)}$ with $\alpha^{(q)} > \alpha^{(q+1)}$ for all $q \in \{0, \ldots, Q-1\}$ and $\alpha^{(Q)} = 1$. This sequence constitutes the update schedule of the discretisation parameter; if the algorithm does not terminate early, eventually the last element of the schedule is $\alpha^{(Q)} = 1$ and one resorts to solving the original DBAP problem without a coarser time discretisation. At each iteration of the algorithm, the following three cases can happen:

1. If $\text{DBAP}^2_{\alpha_2}(\underline{M}) = \text{False}$ and $\text{DBAP}^1_{\alpha_1}(\bar{M}) = \text{True}$, we know that $\text{DBAP}(\underline{M}) = \text{False}$ and $\text{DBAP}(\bar{M}) = \text{True}$. Therefore, we can update the interval $[\underline{z}, \bar{z}]$ as $[\underline{M}, z_{\text{DBAP}^1_{\alpha_1}(\bar{M})}]$ and proceed to the next iteration.
2. If $\text{DBAP}^2_{\alpha_2}(\underline{M}) = \text{DBAP}^1_{\alpha_1}(\bar{M}) = \text{False}$, we know that $\text{DBAP}(\underline{M}) = \text{False}$, but $\text{DBAP}(\bar{M})$ can be True or False. To proceed, we must determine which of the two cases occurs. To this end, we increase the precision of the time discretisation. Let the current discretisation parameter be

$\alpha_1 = \alpha^{(q)}$ for some $q \in \{0, \dots, Q-1\}$. Then, we update it as $\alpha_1 \leftarrow \alpha^{(q+1)}$ and solve $\text{DBAP}^1_{\alpha_1}(\bar{M})$ again. We keep decreasing the value of α_1 following the schedule until either $\text{DBAP}^1_{\alpha_1}(\bar{M}) = \text{True}$ or $\alpha_1 = \alpha^{(Q)} = 1$ and $\text{DBAP}^1_{\alpha_1}(\bar{M}) = \text{False}$. In the first case, we update the interval $[\underline{z}, \bar{z}]$ as $[\underline{M}, z_{\text{DBAP}^1_{\alpha_1}(\bar{M})}]$. In the second case, we update it as $[\bar{M}, \bar{z}]$.

3. If $\text{DBAP}^2_{\alpha_2}(\underline{M}) = \text{DBAP}^1_{\alpha_1}(\bar{M}) = \text{False}$, we know that $\text{DBAP}(\bar{M}) = \text{True}$, but $\text{DBAP}(\underline{M})$ can be True or False. Again, to proceed, we must determine which of the two cases occurs. Let the current discretisation parameter be $\alpha_2 = \alpha^{(q)}$ for some $q \in \{0, \dots, Q-1\}$. Then, we update $\alpha_2 \leftarrow \alpha^{(q+1)}$ and solve $\text{DBAP}^2_{\alpha_2}(\underline{M})$ again. Eventually, we either reach a value of α_2 for which $\text{DBAP}^2_{\alpha_2}(\underline{M}) = \text{False}$, or $\alpha_2 = \alpha^{(Q)} = 1$ and $\text{DBAP}^2_{\alpha_2}(\underline{M}) = \text{True}$. In the first case, we update the interval $[\underline{z}, \bar{z}]$ as $[\underline{M}, z_{\text{DBAP}^1_{\alpha_1}(\bar{M})}]$. In the second case, we update it as $[\underline{z}, z_{\text{DBAP}(\underline{M})}]$. We remark that we can compute $z_{\text{DBAP}(\underline{M})}$ because, when $\alpha_2 = 1$, solving $\text{DBAP}^2_{\alpha_2}(\underline{M})$ corresponds to solving the original $\text{DBAP}(\underline{M})$.

5 Results

This section describes the data that was used for the computational experiments (Section 5.1) and the main results obtained (Section 5.2). Detailed computational results are reported in Appendix B.

5.1 Data

In this work, we introduce and make available two new sets of instances generated from real-world data. We consider two quays in Barcelona's port: 24B (APM Terminals) and 36A (BEST: Barcelona Europe South Terminal). We obtained data on ship movements at these two quays during 2024 from the Port of Barcelona through the open data initiative (Port de Barcelona 2024). Quay 24B is divided into 23 berths, and 36A into 31; however, analysing the data, we determined that only $m = 20$ berths are used in quay 24B and $m = 29$ in quay 36A. For both terminals, all berths have the same length λ.

For each ship i, the dataset indicates the leftmost (σ_i) and rightmost (τ_i) berths used by the ship. From these values, we compute the number of berths required to handle the ship as $\hat{\ell}_i = \tau_i - \sigma_i + 1$. The dataset also directly reports the ship length L_i in metres; however, for operational reasons, a ship can require more than $\lceil L_i/\lambda \rceil$ berths. Indeed, the data show that more than half of the ships require one additional berth, and, in some cases, two or even three extra berths are used. Therefore, we normalise the berth length as $\lambda = 1$ and use $\ell_i = \hat{\ell}_i$ as the ship length, because this quantity reflects the real number of berths used to handle the ship.

The data also record the periods $\underline{\theta}_i$ and $\bar{\theta}_i$ when the ship moored and left the quay. From this data, we compute the handling time $h_i = \bar{\theta}_i - \underline{\theta}_i + 1$. The dataset does not contain the ship arrival times. If we used the mooring time as the arrival time ($a_i = \underline{\theta}_i$), it would be trivial to find the BAP optimal solution by using the real-life mooring time $\underline{\theta}_i$ and berth τ_i. Because such a solution was implemented in practice, it would be feasible, and because each ship moors at the earliest possible time, it would be optimal. Figure 5 shows two such berthing plans, relative to weeks six and seven of 2024, at the BEST terminal (quay 36A).

To obtain challenging instances, then, we proceed as follows. For each week w from the second to the 50^{th}, we create an instance using data from weeks w and $w+1$. We do not use the first and last weeks of the year to avoid spill-overs with years 2023 and 2025, respectively. In each instance, we use data from week w as-is, but we move backwards in time by one week the arrival times of all ships mooring during week $w+1$. In this way, we create a busy week, with roughly twice as many ships as arrive at Barcelona's port during a typical week. Figure 6 shows the result of this procedure on the two weeks depicted in Figure 5. As shown by the many overlaps, a trivial berthing plan that allows each ship to berth as soon as it arrives would be infeasible for the generated instance.

The resulting dataset contains one instance for each quay and each week from the second to the 50^{th}, i.e., $2 \times 49 = 98$ instances. The corresponding problem type is "hyb|dyn|fix|max(compl)". To ensure

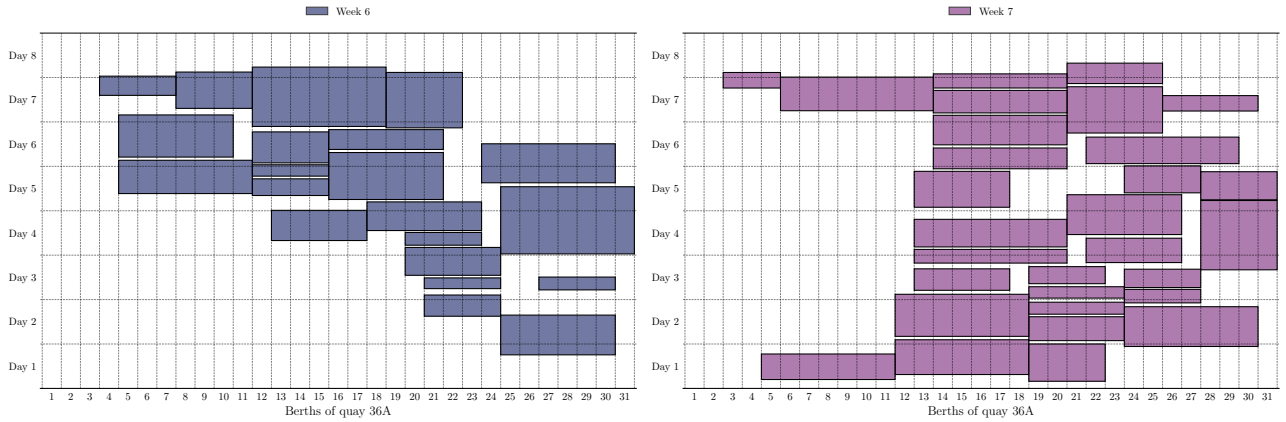

Figure 5: Real-life berthing plans for weeks six (left) and seven (right) of 2024 at the BEST terminal of Barcelona's port. Note that the two leftmost berths are always unused according to the data.

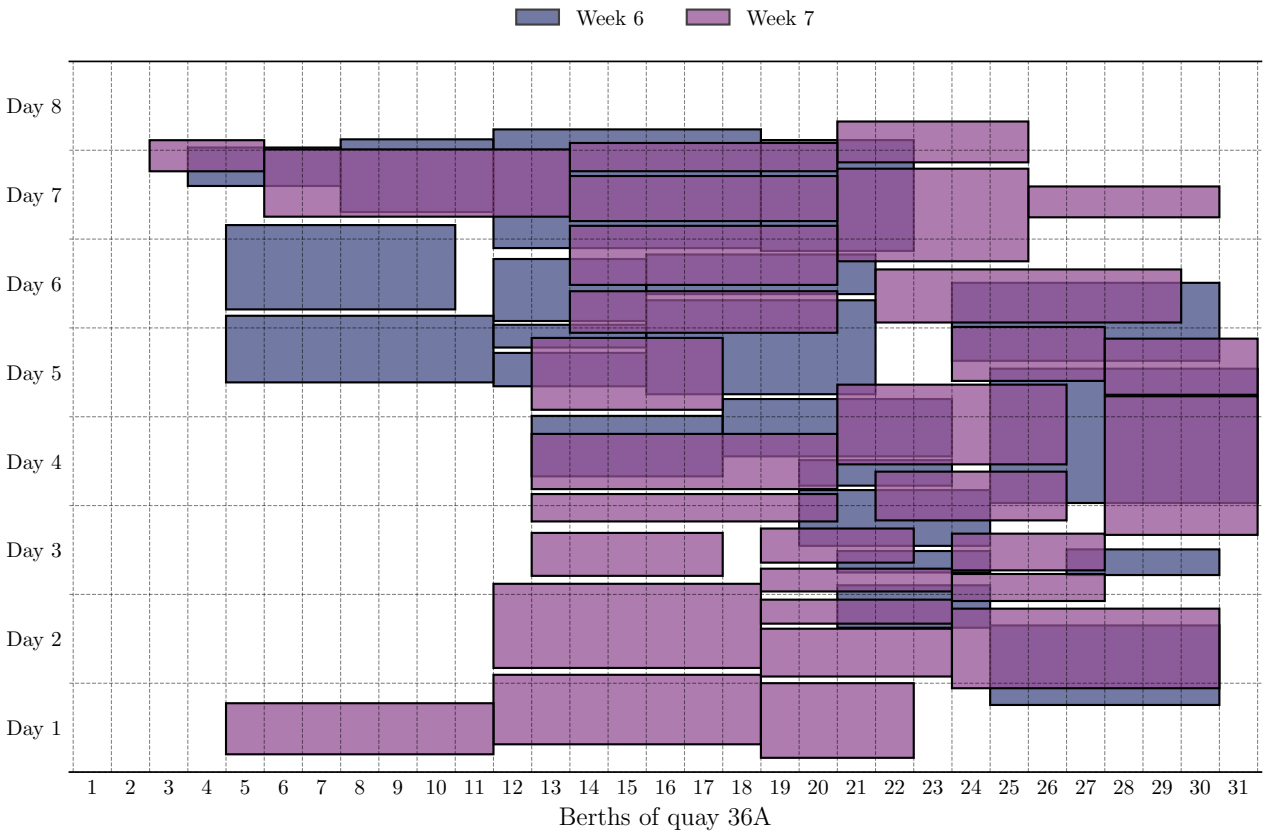

Figure 6: Instance resulting from considering weeks six and seven together, and moving the arrival times of the ships that moored during week seven backwards by one week. The resulting BAP instance can no longer be trivially solved to optimality by letting each ship berth as soon as it arrives.

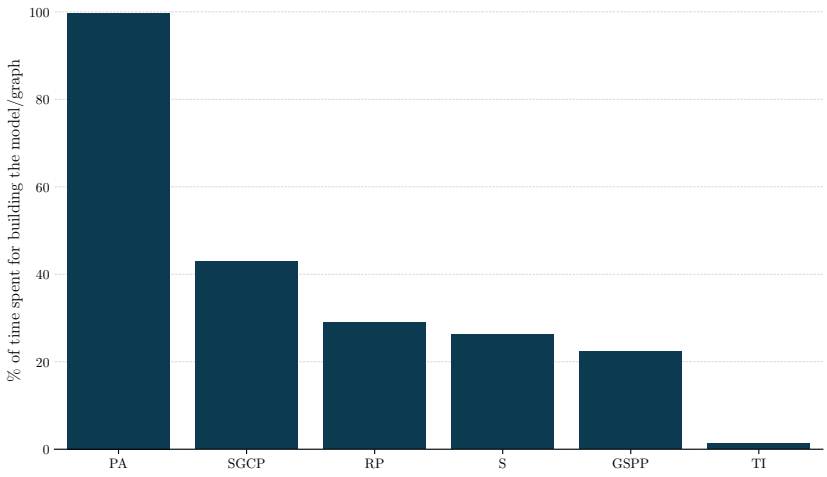

Figure 7: Percentage of the total runtime spent building the mixed-integer models (for formulations RP, PA, S, and TI) or the graphs (for approach SGCP).

feasibility, we consider a time horizon of two weeks, discretised by the hour. The instances and the best known solutions are available on GitHub (Santini 2025).

To summarise:

- We generate 98 instances; 49 refer to quay 24B and 49 to quay 36A of the port of Barcelona, Spain.
- The 24B instances involve $m = 20$ berths and the 36A instances involve $m = 29$ quays.
- The 24B instances involve between $n = 12$ and $n = 29$ ships (median value: 24); the 36A instances involve between $n = 27$ and $n = 59$ ships (median value: 39).
- All instances contain $24 \times 14 = 336$ periods, i.e., a two-week time horizon discretised by the hour.

5.2 Computational results

We present the results obtained comparing our proposed approach with five problem formulations from the literature. The base formulations used for this analysis are the relative position (RP) and position assignment (PA) formulations of Guan and Cheung (2004), the generalised set partitioning problem (GSPP) formulation of Buhrkal et al. (2011), and the sequence variables (S) and time index (TI) models proposed by Ernst et al. (2017). Whenever these formulations could not solve our "hyb|dyn|fix|max(compl)" problem, they were adapted accordingly. All five formulations are presented in this paper's notation (with the eventual adaptations) in Appendix A. In the rest of this section, our proposed approach is denoted "SGCP".

The computations were carried out on a cluster equipped with Intel Xeon processors running at 3.4GHz, and 32GB of memory. Computations were single-threaded, and we imposed a maximum runtime of one hour. For our proposed approach, the runtime includes both building the graphs $(V^{\text{BAP}}, E^{\text{BAP}})$ and running the SGCP-based binary search algorithm. For the formulations, the runtime includes both model construction and solution with Gurobi 12.0.

All formulations except PA produced at least one feasible solution for all instances within the one-hour time limit. Indeed, building the PA model in Python using the `gurobipy` library is extremely time-consuming and was completed within the time limit for only 10 instances. If the model could be built within one hour, its solution was usually quicker (min: 8.99 seconds, max: 537.12 seconds, median: 27.02 seconds) and, for all these ten instances, PA produced a provably optimal solution.

Figure 7 shows the percentage of the total runtime spent building the mixed-integer models (for formulations RP, PA, S, and TI) or the graphs (for approach SGCP). Formulation PA timed out in 88

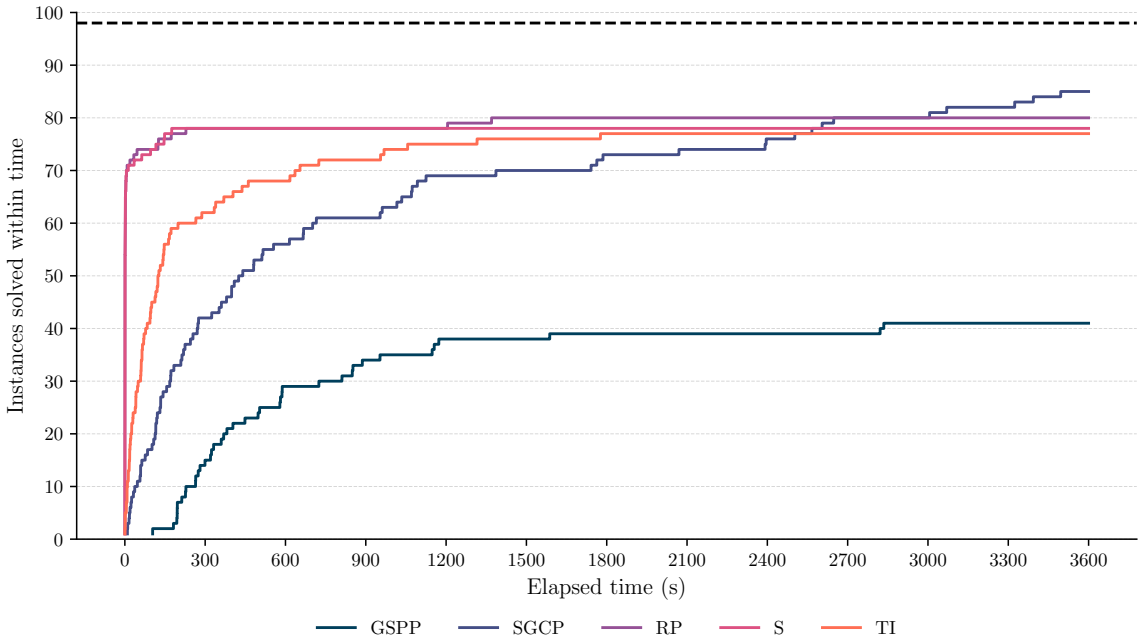

Figure 8: Performance profile for the SGCP-based method and the four formulations GSPP, RP, S, and TI. The x axis reports the elapsed runtime, and the y axis reports the number of instances solved to optimality within a given runtime.

of the 98 instances, and in all timeout cases, the model-building procedure took 100% of the time, i.e., the model was not fully built within one hour. Our proposed SGCP-based approach spends more than 40% of the runtime building the graphs $(V^{\text{BAP}}, E^{\text{BAP}})$ required to solve the decision version of the problem. Note that these are multiple graphs because the vertex and edge sets change with the time discretisation. Formulations RP, S, and GSPP all use between 20% and 30% of their runtime to build the corresponding models, whereas formulation TI uses only 1.28%.

Because formulation PA could only find a feasible solution for 10 out of 98 instances, we exclude this formulation from the rest of our analysis.

Figure 8 shows a performance profile for the SGCP-based method and the four formulations GSPP, RP, S, and TI. The x axis reports the elapsed runtime, and the y axis reports the number of instances solved to optimality within a given runtime. The figure shows that the GSPP formulation is clearly inferior to the other four approaches. This result is interesting because this formulation is the state of the art for the "$\sum$w(wait+hand)" objective function. Our computational experiments show that the formulation is much weaker when adapted to tackle the "max(comp)" objective function.

Furthermore, the other formulations (and, in particular, S and RP) are particularly effective at solving smaller, easier instances. If we restrict ourselves to results obtained within the first 2400 seconds, all three formulations (S, RP, and TI) outperform the SGCP-based method by solving more instances. However, for larger and harder instances, these formulations have a harder time making use of the additional compute time, and they do not significantly increase the number of instances solved to optimality as time increases.

By contrast, the SGCP-based method is slower on small, easy instances, but it uses the available runtime more efficiently by increasing the number of solved instances more consistently. Indeed, for a full runtime of 3600 seconds, the SGCP-based method we propose solves more instances (85) than any formulation. One reason why the SGCP-based method is slower for small instances is that it involves expensive graph creation procedures (which must be repeated for each value of the time discretisation factor). Moreover, remark that, even in the best-case scenario in which the values for α, $\underline{M}$ and $\bar{M}$ are optimal from the beginning (i.e., $\alpha = 1$, $\underline{M} = z^* - 1$ and $\bar{M} = z^*$), we still have to solve two SGCPs: one to prove that $\text{DBAP}(\underline{M}) = \text{False}$, and one to prove that $\text{DBAP}(\bar{M}) = \text{True}$. Therefore,

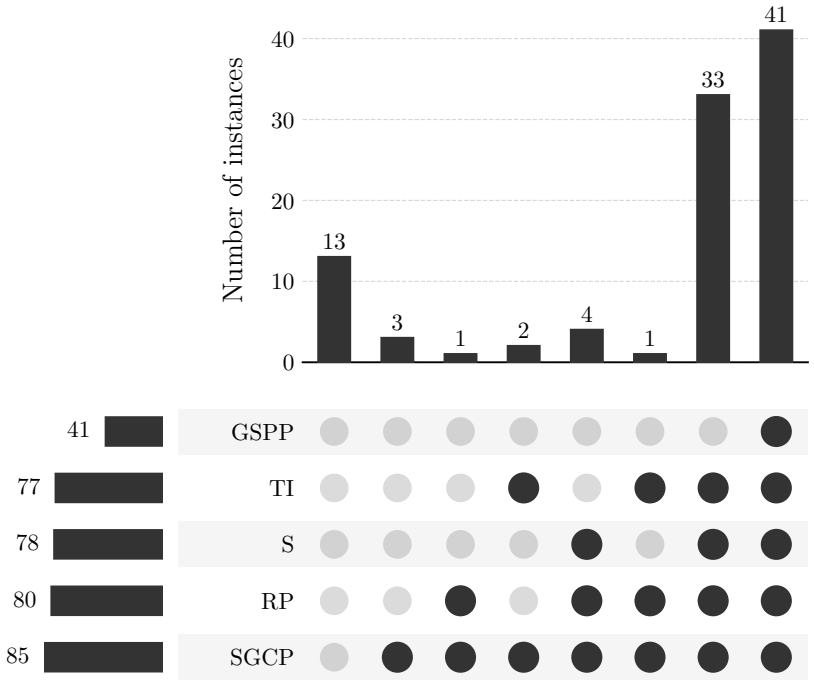

Figure 9: UpSet graph showing how many instances have been solved to optimality by each subset of solvers and by each solver individually.

the SGCP-based method incurs "fixed costs" that make it less attractive for smaller instances, which can be efficiently solved directly with a compact mixed-integer formulation. For larger instances, however, the availability of a fast, state-of-the-art SGCP solver pays off, and the upper and lower bounds can be quickly tightened until the optimal solution is identified.

Figure 9 shows an UpSet plot (Lex et al. 2014) describing how many instances have been solved to optimality by each subset of solvers and by each solver individually. The plot contains three parts. In the bottom-right part, there is a matrix of empty or full circles. Its rows are indexed by the solvers (the SGCP-based method and formulations GSPP, TI, S, and RP), and the columns correspond to subsets of the solvers. A full circle means that the solver is included in the subset, and an empty circle means that the solver is not included in the subset. The top part is a bar chart with one bar for each subset defined in the matrix. The bar height is the number of instances that have been solved to optimality by all the solvers included in the subset. For example, the leftmost bar reports that 13 instances have not been optimally solved by any solver (and, therefore, they remain open). Conversely, the rightmost bar reports that 41 instances have been solved by all solvers, and the bar to its left reports that 33 instances have been solved by all solvers except the GSPP formulation. Finally, the bar chart in the bottom-left part reports the number of instances solved to optimality by each solver.

The SCGP-based approach optimally solves 85 out of 98 instances. The RP, S, and TI formulations solve a slightly lower number of instances (between 77 and 80). The GSPP formulation could only solve 41 instances. Three instances (`24B_22`, `36A_8`, and `36A_40`) could only be solved to proven optimality by the SGCP-based approach. For instance `24B_22`, formulations RP, TI, and S could also find the optimal solution but could not prove its optimality. For the other two instances, no formulation found a primal solution with the same makespan as the optimal solution found by the SGCP-based approach.

6 Conclusions

In this work, we have introduced a new method to solve several variants of the Berth Allocation Problem (BAP), with a particular focus on the "hyb|dyn|fix|max(compl)" problem type, according to the classification scheme of Bierwirth and Meisel (2015). Our method relies on quickly solving

the decision version of this problem via a transformation to the Selective Graph Colouring Problem (SGCP). To speed up the method, we devised a dynamic time discretisation technique that reduces the size of the underlying SGCP graph. We compared our method with five formulations from the literature on instances with 12–59 ships, 20–29 quays and 336 time periods, and found that:

- The PA formulation of Guan and Cheung (2004) is unsuitable to solve instances of this size. Most of the time, it is impossible even to build the model within an hour.
- The GSPP formulation of Christensen and Holst (2008) and Buhrkal et al. (2011), which is the state of the art for other BAP problem types, is not very effective on the studied "hyb|dyn|fix|max(compl)" problem type.
- The other three formulations (RP by Guan and Cheung (2004), and S and TI by Ernst et al. (2017)) are competitive. The RP and S formulations, in particular, can find optimal solutions to smaller instances in short computing times. However, all three formulations struggle with larger, harder instances and do not make effective use of the 1-hour time limit. Indeed, in most cases, they either solve an instance to optimality within the first 30 minutes or leave it unsolved at the end of the 1-hour time limit.
- By contrast, our proposed method is slower than the RP, S, and TI formulations for the smaller, easier instances. However, it utilises computation time more effectively and continues solving more instances within the 1-hour time limit. Indeed, the proposed SGCP-based method solves more instances than any formulation (85 out of 98), and three instances are solved to optimality only by our method.

An important direction for future research is to generalise our approach to BAPs with other objective functions. As mentioned in Section 1, a common variant of the problem asks to minimise the weighted flow time. The decision version of this problem asks whether it is possible to devise a berthing plan with a weighted flow time no greater than a given threshold. However, to our knowledge, the corresponding weighted version of the SGCP has not yet been studied in the literature.

Some authors have considered tactical (Giallombardo et al. 2010) and even strategic (Wawrzyniak et al. 2020) versions of the BAP, with time horizons of up to months. Although decomposition methods are the de facto standard when solving other types of large combinatorial optimisation problems (Chicano et al. 2011; Karimi-Mamaghan et al. 2022; Torkaman et al. 2022; Santini, Schneider, et al. 2023), we could not find any decomposition approach for large-scale BAPs. Such an approach could use the SGCP transformation of the DBAP to detect and focus on small portions of the port's geography or the time horizon for which DBAP is locally infeasible.

Finally, we note that we make our instances, the best known solutions, and the instance-generation code available (Santini 2025). The instances can be used as-is for other BAP problem types, and the generator can be easily adapted to create instances of even more problem types. Furthermore, the generation procedure described in Section 5.1 can be easily modified to create larger instances. Instances with more ships and a longer time horizon can be obtained by merging more than two weeks; instances with more berths can be obtained by merging the 24B and 36A quays.

References

Bierwirth, Christian and Frank Meisel (2015). "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals". In: *European Journal of Operational Research* 244, pp. 675–689. DOI: 10.1016/j.ejor.2014.12.030.

Błażewicz, Jacek, Tai Chiu Cheng Cheng, Maciej Machowiak, and Ceyda Oğuz (2011). "Berth and quay crane allocation: a moldable task scheduling model". In: *Journal of the Operational Research Society* 62 (7), pp. 1189–1197. DOI: 10.1057/jors.2010.54.

Buhrkal, Katja, Sara Zuglian, Stefan Ropke, Jesper Larsen, and Richard Lusby (2011). "Models for the discrete berth allocation problem: A computational comparison". In: *Transportation Research Part E: Logistics and Transportation Review* 47 (4), pp. 461–473. DOI: 10.1016/j.tre.2010.11.016.

Cahyono, Rully Tri, Saskia Puspa Kenaka, and Bayu Jayawardhana (2022). "Simultaneous Allocation and Scheduling of Quay Cranes, Yard Cranes, and Trucks in Dynamical Integrated Container Terminal Operations". In: *IEEE Transactions on Intelligent Transportation Systems* 23 (7), pp. 8564–8578. DOI: 10.1109/TITS.2021.3083598.

Cheong, Chun Yew, Kay Chen Tan, Dikai Liu, and Chiuhsiang Joe Lin (2010). "Multi-objective and prioritized berth allocation in container ports". In: *Annals of Operations Research* 180, pp. 63–103. DOI: 10.1007/s10479-008-0493-0.

Chicano, Francisco, Whitley Darrell, and Enrique Alba (2011). "A Methodology to Find the Elementary Landscape Decomposition of Combinatorial Optimization Problems". In: *Evolutionary Computation* 19 (4), pp. 597–637. DOI: 10.1162/EVCO_a_00039.

Christensen, Clement Gram and Cecilie Terese Holst (2008). "Allokering af kajplads i containerhavne". Thesis number Imm-m.sc.-2008-37. MA thesis. Danish Technical University. URL: https://findit.dtu.dk/en/catalog/5a02e31d5010df4c820fe078.

Cornaz, Denis, Fabio Furini, Enrico Malaguti, and Alberto Santini (2019). "A note on Selective line-graphs and partition colorings". In: *Operations Research Letters* 47 (6), pp. 565–568. DOI: 10.1016/j.orl.2019.08.005.

Correcher, Juan Francisco, Thomas Van den Bossche, Ramon Alvarez-Valdes, and Greet Vanden Berghe (2019). "The berth allocation problem in terminals with irregular layouts". In: *European Journal of Operational Research* 272 (3), pp. 1096–1108. DOI: 10.1016/j.ejor.2018.07.019.

Demange, Marc, Tınaz Ekim, Bernard Ries, and Cerasela Tanasescu (2015). "On some applications of the selective graph coloring problem". In: *European Journal of Operational Research* 240 (2), pp. 307–314. DOI: 10.1016/j.ejor.2014.05.011.

Emde, Simon, Nils Boysen, and Dirk Briskorn (2014). "The berth allocation problem with mobile quay walls: problem definition, solution procedures, and extensions". In: *Journal of Scheduling* 17, pp. 289–303. DOI: 10.1007/s10951-013-0358-5.

Ernst, Andreas, Ceyda Oğuz, Gaurav Singh, and Gita Taherkhani (2017). "Mathematical models for the berth allocation problem in dry bulk terminals". In: *Journal of Scheduling* 20, pp. 459–473. DOI: 10.1007/s10951-015-0510-8.

Furini, Fabio, Enrico Malaguti, and Alberto Santini (2018). "An exact algorithm for the Partition Coloring Problem". In: *Computers & Operations Research* 92, pp. 170–181. DOI: 10.1016/j.cor.2017.12.019.

Giallombardo, Giovanni, Luigi Moccia, Matteo Salani, and Ilaria Vacca (2010). "Modeling and solving the Tactical Berth Allocation Problem". In: *Transportation Research Part B: Methodological* 44 (2), pp. 232–245. DOI: 10.1016/j.trb.2009.07.003.

Glover, Fred and Manuel Laguna (1997). "Tabu Search". In: *Handbook of Combinatorial Optimization*. Ed. by Ding-Zhu Du and Panos Paradalos. Kluwer Academic Publishers, pp. 621–757.

Grubišić, Neven and Mate Barić (2015). "A Contribution to Berth Allocation Problem Solution with Draft Restrictions". In: *Pomorski zbornik* 49–50 (1). Article ID 138194, pp. 127–142.

Guan, Yongpei and Raymond Kaman Cheung (2004). "The berth allocation problem: models and solution methods". In: *OR Spectrum* 26, pp. 75–92. DOI: 10.1007/s00291-003-0140-8.

Hansen, Pierre and Ceyda Oğuz (May 2003). *A Note on Formulations of Static and Dynamic Berth Allocation Problems*. Tech. rep. G-2003-30. Montréal, Canada: Le Cahiers du GERAD.

Held, Stephan, William Cook, and Edward Clyde Sewell (2012). "Maximum-weight stable sets and safe lower bounds for graph coloring". In: *Mathematical Programming Computation* 4, pp. 363–381. DOI: 10.1007/s12532-012-0042-3.

Hoshino, Edna Ayako, Yuri Abitbol Frota, and Cid Carvalho de Souza (2011). "A branch-and-price approach for the partition coloring problem". In: *Operations Research Letters* 39 (2), pp. 132–137. DOI: 10.1016/j.orl.2011.02.006.

Imai, Akio, Etsuko Nishimura, and Stratos Papadimitriou (2001). "The dynamic berth allocation problem for a container port". In: *Transportation Research Part B: Methodological* 35 (4), pp. 401–417. DOI: 10.1016/S0191-2615(99)00057-0.

Ji, Bin, Han Huang, and Samson Yu (2023). "An Enhanced NSGA-II for Solving Berth Allocation and Quay Crane Assignment Problem With Stochastic Arrival Times". In: *IEEE Transactions on Intelligent Transportation Systems* 24 (1), pp. 459–473. DOI: 10.1109/TITS.2022.3213834.

Karimi-Mamaghan, Maryam, Mehrdad Mohammadi, Patrick Meyer, Amir Mohammad Karimi-Mamaghan, and El-Ghazali Talbi (2022). "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art". In: *European Journal of Operational Research* 296 (2), pp. 393–422. DOI: 10.1016/j.ejor.2021.04.032.

Kramer, Arthur, Eduardo Lalla-Ruiz, Manuel Iori, and Stefan Voß (2019). "Novel formulations and modeling enhancements for the dynamic berth allocation problem". In: *European Journal of Operational Research* 278 (1), pp. 170–185. DOI: 10.1016/j.ejor.2019.03.036.

Lalla-Ruiz, Eduardo, Christopher Expósito-Izquierdo, Belén Melián-Batista, and José Marcos Moreno-Vega (2016). "A Set-Partitioning-based model for the Berth Allocation Problem under Time-Dependent Limitations". In: *European Journal of Operational Research* 250 (3), pp. 1001–1012. DOI: 10.1016/j.ejor.2015.10.021.

Lalla-Ruiz, Eduardo, Belén Melián-Batista, and José Marcos Moreno-Vega (2012). "Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem". In: *Engineering Applications of Artificial Intelligence* 25 (6), pp. 1132–1141. DOI: 10.1016/j.engappai.2012.06.001.

Lee, Der-Horng and Jiang Hang Chen (2010). "An Improved Approach for Quay Crane Scheduling with Non-Crossing Constraints". In: *Engineering Optimization* 42 (1), pp. 1–15. DOI: 10.1080/03052150902943020.

Lee, Der-Horng and Hui Qiu Wang (2010). "Integrated discrete berth allocation and quay crane scheduling in port container terminals". In: *Engineering Optimization* 42.8, pp. 747–761. DOI: 10.1080/03052150903406571.

Lex, Alexander, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister (2014). "UpSet: Visualization of Intersecting Sets". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12, pp. 1983–1992. DOI: 10.1109/TVCG.2014.2346248.

Li, Chung-Lun, Xiaoqiang Cai, and Chung-Yee Lee (1998). "Scheduling with multiple-job-on-one-processor pattern". In: *IIE Transactions* 30.5, pp. 433–445. DOI: 10.1080/07408179808966484.

Li, Ying, Feng Chu, Feifeng Zheng, and Ming Liu (2022). "A Bi-Objective Optimization for Integrated Berth Allocation and Quay Crane Assignment With Preventive Maintenance Activities". In: *IEEE Transactions on Intelligent Transportation Systems* 23 (4), pp. 2938–2955. DOI: 10.1109/TITS.2020.3023701.

Lim, Andrew (1988). "The berth planning problem". In: *Operations Research Letters* 22, pp. 105–110. DOI: 10.1016/s0167-6377(98)00010-8.

Liu, Ming, Rongfan Liu, Feng Chu, and Chengbin Chu (July 2018). "Mathematical Model and Solution Approach for Berth Allocation Problem in Tidal Bulk Ports with Different Vessel Draft Requirements". In: *Proceedings of the IEEE 15th International Conference on Service Systems and Service Management*. Hangzhou, China, pp. 1–6. DOI: 10.1109/ICSSSM.2018.8465036.

Malaguti, Enrico, Silvano Martello, and Alberto Santini (2018). "The traveling salesman problem with pickups, deliveries, and draft limits". In: *Omega* 74, pp. 50–58. DOI: 10.1016/j.omega.2017.01.005.

Monaco, Maria Flavia and Marcello Samarra (2007). "The Berth Allocation Problem: A Strong Formulation Solved by a Lagrangean Approach". In: *Transportation Science* 41 (2), pp. 265–280. DOI: 10.1287/trsc.1060.0171.

Nishi, Tatsushi, Tatsuya Okura, Eduardo Lalla-Ruiz, and Stefan Voß (2020). "A dynamic programming-based matheuristic for the dynamic berth allocation problem". In: *Annals of Operations Research* 286, pp. 391–410. DOI: 10.1007/s10479-017-2715-9.

Port de Barcelona (2024). *Open Data*. URL: https://opendata.portdebarcelona.cat/en/.

Ropke, Stefan and David Pisinger (2006). "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows". In: *Transportation Science* 40.4, pp. 455–472. DOI: 10.1287/trsc.1050.0135.

Santini, Alberto (Apr. 2017). "Combinatorial Optimisation Problems in Logistics and Scheduling". PhD thesis. Alma Mater Studiorum Università di Bologna. URL: http://amsdottorato.unibo.it/7908/.

Santini, Alberto (2025). *Instances, Best Known Solutions and Models for Berth Allocation Problems*. DOI: 10.5281/zenodo.17665596. URL: https://github.com/alberto-santini/berth-allocation-problems/.

Santini, Alberto, Henrik Alsing Friberg, and Stefan Ropke (2015). "A note on a model for quay crane scheduling with non-crossing constraints". In: *Engineering Optimization* 47 (6), pp. 860–865. DOI: 10.1080/0305215X.2014.958731.

Santini, Alberto, Stefan Ropke, and Lars Magnus Hvattum (2018). "A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic". In: *Journal of Heuristics* 24, pp. 783–815. DOI: 10.1007/s10732-018-9377-x.

Santini, Alberto, Michael Schneider, Thibaut Vidal, and Daniele Vigo (2023). "Decomposition Strategies for Vehicle Routing Heuristics". In: *INFORMS Journal on Computing* 35.3, pp. 543–559. DOI: 10.1287/ijoc.2023.1288.

Torkaman, Somayeh, Mohammad Reza Akbari Jokar, Nevin Mutlu, and Tom Van Woensel (2022). "Rolling horizon-based heuristics for solving a production-routing problem with price-dependent demand". In: *Computers & Operations Research* 148. Article ID 105973. DOI: 10.1016/j.cor.2022.105973.

Vacca, Ilaria, Matteo Salani, and Michel Bierlaire (2012). "An Exact Algorithm for the Integrated Planning of Berth Allocation and Quay Crane Assignment". In: *Transportation Science* 47.2. DOI: 10.1287/trsc.1120.0428.

Wawrzyniak, Jakub, Maciej Drozdowski, and Éric Sanlaville (2020). "Selecting algorithms for large berth allocation problems". In: *European Journal of Operational Research* 283 (3), pp. 844–862. DOI: 10.1016/j.ejor.2019.11.055.

Zhu, Enqiang, Fei Jiang, and Jin Liu Chanjuan amd Xu (2022). "Partition Independent Set and Reduction-Based Approach for Partition Coloring Problem". In: *IEEE Transactions on Cybernetics* 52 (6), pp. 4960–4969. DOI: 10.1109/TCYB.2020.3025819.

A Mathematical models

In this section, we describe the five formulations we use to assess the quality of our proposed approach. We describe these formulations using the notation introduced in this paper and highlight any significant modifications of the original models.

A.1 Model "RP" of Guan and Cheung (2004)

This formulation was originally presented for a "hyb|dyn|fix|$\sum$w(wait+hand)" BAP. Furthermore, the authors assume that all berths have the same length. Therefore, the length of a ship is simply expressed as the number of berths it occupies. We denote this parameter as $\hat{\ell}_i$ ($i \in \{1, \dots, n\}$) and will work under the same assumption.

This model uses the following decision variables. Variables $x_{i_1 i_2} \in \{0, 1\}$ and $y_{i_1 i_2} \in \{0, 1\}$ for each pair of ships i_1, i_2 ($i_1 \neq i_2$). Variable $x_{i_1 i_2}$ takes value one if and only if i_1's rectangle is on the left of i_2's, and there is no overlap. Variable $y_{i_1 i_2}$ takes value one if and only if i_1's rectangle is below i_2's, and there is no overlap. Variables $u_i, c_i \in \{1, \dots, T\}$ for each ship i. Variable u_i records the ship's mooring time, and variable c_i denotes the completion time, i.e., the time when the ship departs from the port. Variable $v_i \in \{1, \dots, m\}$ for each ship i, denoting the leftmost berth used by the ship. Variable $C \geq 0$, the total makespan.

The model adapted for a "hyb|dyn|fix|max(compl)" BAP reads as follows.

$$\min \quad C \tag{4a}$$

$$\text{s.t.} \quad C \geq c_i \qquad \forall i \in \{1, \dots, n\} \tag{4b}$$

$$a_i \leq u_i \leq T - h_i \qquad \forall i \in \{1, \dots, n\} \tag{4c}$$

$$1 \leq v_i \leq m - \hat{\ell}_i + 1 \qquad \forall i \in \{1, \dots, n\} \tag{4d}$$

$$c_i = u_i + h_i \qquad \forall i \in \{1, \dots, n\} \tag{4e}$$

$$u_{i_2} - u_{i_1} - h_{i_1} - T(x_{i_1 i_2} - 1) \geq 0 \qquad \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2 \tag{4f}$$

$$v_{i_2} - v_{i_1} - \hat{\ell}_{i_1} - m(y_{i_1 i_2} - 1) \geq 0 \qquad \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2 \tag{4g}$$

$$x_{i_1 i_2} + x_{i_2 i_1} + y_{i_1 i_2} + y_{i_2 i_1} \geq 1 \qquad \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2 \tag{4h}$$

$$x_{i_1 i_2} + x_{i_2 i_1} \leq 1 \qquad \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2 \tag{4i}$$

$$y_{i_1 i_2} + y_{i_2 i_1} \leq 1 \qquad \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2. \tag{4j}$$

The objective function (4a), together with constraints (4b), minimises the total makespan. Constraints (4c) and (4d) provide bounds for variables u and v. Constraint (4e) links variables c and u. Constraints (4f) and (4g) are no-overlap constraints. Constraints (4h), (4i), and (4j) ensure that the correct mutual position of any two ships is identified. Compared with the original formulation, we introduced variable C and constraints (4b) to allow minimising the total makespan rather than the weighted sum of completion times.

A.2 Model "PA" of Guan and Cheung (2004)

As mentioned above, Guan and Cheung (2004) assume that all berths have the same length. While the RP model cannot be straightforwardly modified to overcome this assumption, the PA model can be amended to account for berths of variable lengths.

In the following, we present the more general model that uses the following variables. Variable $x_{ijt} \in \{0, 1\}$ for each ship i, berth j and period $t \geq a_i$. This variable takes value one if and only if i occupies j during t. Variable $y_{ijt} \in \{0, 1\}$ for each ship i, berth j such that $\delta_j \leq \Lambda - \ell_i$ and period t such that $a_i \leq t \leq T - h_i$. This variable takes value one if and only if i moors at berth j at t; j must then be the leftmost berth used by i. Variable $c_i \in \{1, \ldots, T\}$ for each ship i records the ships' departure times. The variable $C \geq 0$ is the total makespan.

The model reads as follows; $\mathcal{M}$ is a sufficiently large positive number.

$$\min \quad C \tag{5a}$$

$$\text{subject to} \quad C \geq c_i \qquad \forall i \in \{1, \ldots, n\} \tag{5b}$$

$$c_i = \sum_{\substack{j=1 \\ \delta_j < \Lambda - \ell_i}}^{m} \sum_{t=a_i}^{T-h_i} t y_{ijt} + h_i \qquad \forall i \in \{1, \ldots, n\} \tag{5c}$$

$$\sum_{\substack{j=1 \\ \delta_j < \Lambda - \ell_i}}^{m} \sum_{t=a_i}^{T-h_i} y_{ijt} = 1 \qquad \forall i \in \{1, \ldots, n\} \tag{5d}$$

$$\sum_{\substack{j'=j \\ \delta_{j'} - \delta_j < \ell_i}}^{m} \sum_{\substack{t'=t \\ t' - t < h_i}}^{T} x_{ij't'} - h_i \ell_i - \mathcal{M}(y_{ijt} - 1) \geq 0 \qquad \begin{array}{l} \forall i \in \{1, \ldots, n\}, \\ \forall j \in \{1, \ldots, m\},\ \delta_j \leq \Lambda - \ell_i, \\ \forall t \in \{a_i, \ldots, T - h_i\} \end{array} \tag{5e}$$

$$\sum_{i=1}^{n} x_{ijt} \leq 1 \qquad \forall j \in \{1, \ldots, m\},\ \forall t \in \{1, \ldots, T\}. \tag{5f}$$

The objective function (5a) and constraint (5b) are analogous to (4a) and (4b). Constraint (5c) sets the value of variables c. Constraint (5d) ensures that each ship is assigned exactly one mooring berth and time. Constraint (5e) links the x and y variables. Constraint (5f) ensures that no more than one ship uses each berth at any given time.

A.3 Model "GSPP" of Christensen and Holst (2008) and Buhrkal et al. (2011)

This model was originally proposed for a "disc|dyn|pos|$\sum$w(wait+hand)" BAP. However, it can be amended to consider a hybrid spatial attribute and to minimise the makespan. To present the model,

it is useful to remind the definitions of V^{BAP} and E^{BAP} given in Section 3 and define:

$$V_{jt}^{\text{BAP}} = \{(i, j', t') \in V^{\text{BAP}} \ : \ \delta_{j'} \leq \delta_j \leq \delta_{j'} + \ell_i \text{ and } t' \leq t \leq t' + h_i\}.$$

Each set V_{jt}^{BAP} contains all assignments to ships to berths and mooring times that cause berth j to be occupied at time t.

The GSPP model uses variables $x_v \in \{0, 1\}$ defined for each $v \in V^{\text{BAP}}$ and a single variable $C \geq 0$. Variable x_v takes the value one for $v = (i, j, t)$ if ship i moors at berth j at time t. The model reads as follows.

$$\begin{aligned}
\min \quad & C && && \text{(6a)}\\
\text{subject to} \quad & C \geq (t + h_i)x_v && \forall v = (i, j, t) \in V^{\text{BAP}} && \text{(6b)}\\
& \sum_{v=(i,j,t)\in V^{\text{BAP}}} x_v = 1 && \forall i \in \{1, \ldots, n\} && \text{(6c)}\\
& \sum_{v\in V_{jt}^{\text{BAP}}} x_v \leq 1 && \forall j \in \{1, \ldots, m\},\ \forall t \in \{1, \ldots, T\}. && \text{(6d)}
\end{aligned}$$

Constraint (6c) ensures that each ship is assigned a mooring time and berth. Constraint (6d) prevents more than one ship from using a given berth at a given time.

A.4 Model "S" of Ernst et al. (2017)

Ernst et al. (2017) consider a "cont|dyn|fix|$\sum$(compl)" BAP. Indeed, the authors assume the quay is continuous and ships can moor at any point. To adapt this model for a hybrid spatial attribute allowing a ship to occupy several berths, we assume that the quay is divided into equal-length berths and that a ship i occupies $\hat{\ell}_i$ of them.

The model uses the following variables. Variable $x_{i_1 i_2} \in \{0, 1\}$ for each pair of ships i_1, i_2 ($i_1 \neq i_2$). This variable takes value one if and only if ship i_1 completes before ship i_2 starts loading/unloading operations. Variable $I_{i_1 i_2} \in \{0, 1\}$ for each pair of ships i_1, i_2 ($i_1 \neq i_2$). This variable takes value one if and only if the ships overlap in time, and i_1 moors on the right of i_2. Variables $s_i, c_i \in \{1, \ldots, T\}$ are the periods when ship i moors and completes operations. Variable s_i is bounded from below by a_i. Variable $y_i \in \{1, \ldots, m\}$ is the leftmost berth used by ship i. This variable is bounded from above by $m - \hat{\ell}_i$. $C \geq 0$ is the total makespan.

In the following, we present the model together with strengthenings and valid inequalities introduced in (Ernst et al. 2017). To this end, we assume that $\prec$ is an arbitrary ordering of the ships such that if $i_1 \prec i_2$, then $a_{i_1} \leq a_{i_2}$.

$$\begin{aligned}
\min \quad & C && && \text{(7a)}\\
\text{subject to} \quad & C \geq c_i && \forall i \in \{1, \ldots, n\} && \text{(7b)}\\
& c_i = s_i + h_i && \forall i \in \{1, \ldots, n\} && \text{(7c)}\\
& x_{i_1 i_2} + x_{i_2 i_1} + I_{i_1 i_2} + I_{i_2 i_1} = 1 && \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2 && \text{(7d)}\\
& s_{i_2} \geq c_{i_1} - \mathcal{M}(1 - x_{i_1 i_2}) && \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2 && \text{(7e)}\\
& y_{i_2} \geq \hat{\ell}_{i_1} I_{i_1 i_2} + y_{i_1} - (m - \hat{\ell}_{i_1})(1 - I_{i_1 i_2}) && \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \neq i_2 && \text{(7f)}\\
& \sum_{\substack{i_2=1\\ \hat{\ell}_{i_1}+\hat{\ell}_{i_2}>m}}^{n} I_{i_1 i_2} = 0 && \forall i_1 \in \{1, \ldots, n\} && \text{(7g)}\\
& c_{i_1} \leq c_{i_2} \text{ and } x_{i_2 i_1} = 0 && \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \prec i_2,\ h_{i_1} = h_{i_2},\ \hat{\ell}_{i_1} = \hat{\ell}_{i_2} && \text{(7h)}\\
& x_{i_2 i_1} \leq \sum_{\substack{i_3=1\\ i_3\neq i_1, i_2}}^{n} (I_{i_3 i_1} + I_{i_1 i_3}) && \forall i_1, i_2 \in \{1, \ldots, n\},\ i_1 \prec i_2,\ h_{i_1} = h_{i_2},\ \hat{\ell}_{i_1} < \hat{\ell}_{i_2} && \text{(7i)}
\end{aligned}$$

$$s_{i_1} \geq \frac{1}{\Lambda}\left(\min_{i_2 \in \{1,\ldots,n\}} a_{i_2} + \sum_{\substack{i_2=1 \\ i_2 \neq i_1}}^{n} h_{i_2}\hat{\ell}_{i_2} x_{i_2 i_1}\right) \qquad \forall i_1 \in V \tag{7j}$$

$$s_{i_1} \geq \min_{i_2 \in \{1,\ldots,n\}} a_{i_2} + \sum_{\substack{i_2=1 \\ i_2 \neq i_1 \\ (k+1)\hat{\ell}_{i_2} > m}}^{n} \frac{h_{i_2}}{k} x_{i_2 i_1} \qquad \forall i_1 \in \{1,\ldots,n\},\ \forall k \in \{1,2,3\} \tag{7k}$$

$$y_{i_1} \leq \sum_{\substack{i_2=1 \\ i_2 \neq i_1}}^{n} \hat{\ell}_{i_2} I_{i_2 i_1} \qquad \forall i_1 \in \{1,\ldots,n\}. \tag{7l}$$

The objective function (7a) and constraints (7b) are used to minimise the makespan. By contrast, the original model in (Ernst et al. 2017) minimises the sum of the completion times. Constraint (7c) links the c and s variables. Constraint (7d) are no-overlap constraints, constraint (7e) link the s and x variables, while (7f) link the y and I variables. All subsequent constraints are valid inequalities that Ernst et al. (2017) add to improve the quality of the continuous relaxation of the model. Constraint (7g) forbids handling simultaneously a pair of ships that are too long to fit in the quay. Constraints (7h) and (7i) use the ordering $\prec$ to break some symmetry of the formulation. Constraint (7j) computes the minimum duration for which a ship's rectangle must overlap another rectangle of area $h_{i_2} \times \hat{\ell}_{i_2}$. Constraint (7k) is a no-overlap constraint for ships of length larger than $m/(k+1)$ for a few small values of k. Constraint (7l) is an additional symmetry-breaking constraint.

A.5 Model "TI" of Ernst et al. (2017)

This model uses the same variables as model S, with the additional $q_{it} \in \{0,1\}$ taking value 1 iff ship i completes by time t. The model reads as follows.

$$\min \quad C \tag{8a}$$

$$\text{subject to} \quad \text{(7b), (7c), (7d), (7e), (7f)}$$

$$c_i = \sum_{t=a_i+h_i}^{T-1} t\big(q_{i,t+1} - q_{it}\big) \qquad \forall i \in \{1,\ldots,n\} \tag{8b}$$

$$q_{i,t+1} \geq q_{it} \qquad \forall i \in \{1,\ldots,n\},\ \forall t \in \{a_i + h_i, \ldots, T-1\} \tag{8c}$$

$$q_{iT} = 1 \qquad \forall i \in \{1,\ldots,n\} \tag{8d}$$

$$\sum_{\substack{i=1 \\ t+h_i \leq T}}^{n} \hat{\ell}_i\big(q_{i,t+h_i} - q_{it}\big) \leq \Lambda \qquad \forall t \in \{1,\ldots,T\}. \tag{8e}$$

Constraint (8b) links the c and q variables, setting the ship completion time c_i as the period when the value of variable q changes from zero to one. Constraint (8c) imposes that variables are non-decreasing in t, and constraint (8d) ensures that all ships have completed by the end of the time horizon. Finally, constraint (8e) ensures that all the ships moored at any given time fit in the quay.

B Detailed results

This appendix reports detailed computational results. Table 2 reports the results relative to the SGCP-based method we propose. Tables 3 to 7 report the results relative to the RP, PA, GSPP, S, and TI formulations, respectively.

Instance	Primal bound	Dual bound	Total time
24B_2	162	162	115.7
24B_3	164	164	209.2
24B_4	169	169	270.4
24B_5	173	173	63.4
24B_6	189	189	408.7
24B_7	180	180	84.7
24B_8	161	161	171.1
24B_9	161	161	172.1
24B_10	165	165	272.8
24B_11	203	203	224.7
24B_12	193	193	183.1
24B_13	174	174	213.3
24B_14	176	176	10.6
24B_15	176	176	32.0
24B_16	199	199	58.8
24B_17	172	172	15.6
24B_18	166	166	352.1
24B_19	186	186	254.4
24B_20	200	200	107.7
24B_21	192	192	360.6
24B_22	186	186	3070.0
24B_23	169	169	76.0
24B_24	168	168	46.7
24B_25	161	161	22.9
24B_26	192	192	440.5
24B_27	183	183	113.0
24B_28	191	191	133.7
24B_29	171	171	118.9
24B_30	177	177	115.3
24B_31	197	197	398.7
24B_32	171	171	25.2
24B_33	183	183	481.0
24B_34	200	187	3600.0
24B_35	185	185	59.0
24B_36	173	173	245.0
24B_37	171	171	9.5
24B_38	182	182	56.6
24B_39	191	191	134.0
24B_40	176	176	17.2
24B_41	189	189	101.6
24B_42	175	175	167.7
24B_43	180	180	1741.4
24B_44	185	185	953.6
24B_45	185	185	512.0
24B_46	162	162	19.5
24B_47	197	197	379.8
24B_48	181	181	614.8
24B_49	186	186	156.1
24B_50	197	197	121.3

Instance	Primal bound	Dual bound	Total time
36A_2	176	176	701.4
36A_3	173	173	1073.3
36A_4	165	165	2395.6
36A_5	184	184	516.1
36A_6	178	172	3600.0
36A_7	197	175	3600.0
36A_8	178	178	3495.8
36A_9	197	197	555.4
36A_10	208	196	3600.0
36A_11	210	189	3600.0
36A_12	178	178	2604.6
36A_13	186	180	3600.0
36A_14	216	191	3600.0
36A_15	188	168	3600.0
36A_16	200	175	3600.0
36A_17	197	197	1786.0
36A_18	194	194	398.9
36A_19	185	172	3600.0
36A_20	217	178	3600.0
36A_21	198	198	2502.5
36A_22	183	175	3600.0
36A_23	205	184	3600.0
36A_24	198	198	667.5
36A_25	194	194	3006.1
36A_26	173	173	3323.9
36A_27	195	195	1070.6
36A_28	187	187	1016.1
36A_29	171	171	666.2
36A_30	191	191	142.7
36A_31	196	196	275.3
36A_32	180	180	1763.0
36A_33	196	196	1386.3
36A_34	178	178	715.4
36A_35	188	188	36.9
36A_36	197	197	2647.9
36A_37	203	203	425.1
36A_38	193	193	481.9
36A_39	173	173	219.2
36A_40	182	182	3393.6
36A_41	180	180	131.1
36A_42	169	169	2391.6
36A_43	174	174	324.4
36A_44	181	181	2069.8
36A_45	193	193	1034.4
36A_46	185	185	1092.6
36A_47	192	192	961.3
36A_48	178	178	2566.4
36A_49	174	174	3498.4
36A_50	186	186	1125.1

Table 2: Detailed results for the SGCP-based approach.

Instance	Primal bound	Dual bound	Total time
24B_2	162	162	0.0
24B_3	164	164	0.1
24B_4	169	169	0.1
24B_5	173	173	0.2
24B_6	189	189	1.7
24B_7	180	180	0.1
24B_8	161	161	0.1
24B_9	161	161	0.1
24B_10	165	165	0.1
24B_11	203	203	0.1
24B_12	193	193	0.0
24B_13	174	174	0.1
24B_14	176	176	0.1
24B_15	176	176	0.3
24B_16	199	199	0.2
24B_17	172	172	0.1
24B_18	166	166	0.1
24B_19	186	186	5.0
24B_20	200	200	0.1
24B_21	192	192	0.1
24B_22	186	184	3600.0
24B_23	169	169	228.1
24B_24	168	168	0.2
24B_25	161	161	0.1
24B_26	192	192	0.1
24B_27	183	183	2.5
24B_28	191	191	0.2
24B_29	171	171	45.5
24B_30	177	177	0.9
24B_31	197	197	0.1
24B_32	171	171	0.1
24B_33	183	183	0.1
24B_34	202	179	3600.0
24B_35	185	185	1.6
24B_36	173	173	0.1
24B_37	171	171	0.1
24B_38	182	182	0.4
24B_39	191	191	1.0
24B_40	176	176	1.0
24B_41	189	189	3.5
24B_42	175	175	1.5
24B_43	180	179	3600.0
24B_44	185	183	3600.0
24B_45	185	185	0.1
24B_46	162	162	0.1
24B_47	197	197	0.1
24B_48	181	181	0.1
24B_49	186	186	0.7
24B_50	197	197	0.2

Instance	Primal bound	Dual bound	Total time
36A_2	176	176	0.1
36A_3	173	173	33.4
36A_4	165	165	2244.5
36A_5	184	184	1.7
36A_6	178	172	3600.0
36A_7	186	174	3600.0
36A_8	181	178	3600.0
36A_9	197	197	1.9
36A_10	202	196	3600.0
36A_11	202	189	3600.0
36A_12	178	178	173.3
36A_13	187	180	3600.0
36A_14	208	191	3600.0
36A_15	191	168	3600.0
36A_16	197	173	3600.0
36A_17	197	197	1369.9
36A_18	194	194	0.6
36A_19	181	172	3600.0
36A_20	209	178	3600.0
36A_21	198	198	18.9
36A_22	180	175	3600.0
36A_23	192	184	3600.0
36A_24	198	198	0.4
36A_25	194	194	0.3
36A_26	173	173	1205.5
36A_27	195	195	1.9
36A_28	187	187	125.6
36A_29	171	171	6.1
36A_30	191	191	0.2
36A_31	196	196	2.1
36A_32	180	180	123.7
36A_33	196	196	0.2
36A_34	178	178	0.2
36A_35	188	188	0.4
36A_36	197	197	0.1
36A_37	203	203	2.4
36A_38	193	193	8.4
36A_39	173	173	3.6
36A_40	184	176	3600.0
36A_41	180	180	0.4
36A_42	169	169	0.3
36A_43	174	174	0.6
36A_44	181	181	0.2
36A_45	193	193	0.2
36A_46	185	185	0.1
36A_47	192	192	0.2
36A_48	178	178	0.2
36A_49	174	174	0.2
36A_50	186	186	0.2

Table 3: Detailed results for the RP formulation.

Instance	Primal bound	Dual bound	Total time
24B_2	162	162	1851.3
24B_3	164	164	3600.0
24B_4	—	—	3600.0
24B_5	—	—	3600.0
24B_6	—	—	3600.0
24B_7	—	—	3600.0
24B_8	161	161	3600.0
24B_9	161	161	3600.0
24B_10	—	—	3600.0
24B_11	203	203	3600.0
24B_12	193	193	3600.0
24B_13	174	174	3600.0
24B_14	176	176	3600.0
24B_15	—	—	3600.0
24B_16	—	—	3600.0
24B_17	172	172	3600.0
24B_18	—	—	3600.0
24B_19	—	—	3600.0
24B_20	—	—	3600.0
24B_21	—	—	3600.0
24B_22	—	—	3600.0
24B_23	—	—	3600.0
24B_24	—	—	3600.0
24B_25	—	—	3600.0
24B_26	—	—	3600.0
24B_27	—	—	3600.0
24B_28	—	—	3600.0
24B_29	—	—	3600.0
24B_30	—	—	3600.0
24B_31	—	—	3600.0
24B_32	—	—	3600.0
24B_33	—	—	3600.0
24B_34	—	—	3600.0
24B_35	—	—	3600.0
24B_36	—	—	3600.0
24B_37	171	171	3600.0
24B_38	—	—	3600.0
24B_39	—	—	3600.0
24B_40	—	—	3600.0
24B_41	—	—	3600.0
24B_42	—	—	3600.0
24B_43	—	—	3600.0
24B_44	—	—	3600.0
24B_45	—	—	3600.0
24B_46	—	—	3600.0
24B_47	—	—	3600.0
24B_48	—	—	3600.0
24B_49	—	—	3600.0
24B_50	—	—	3600.0

Instance	Primal bound	Dual bound	Total time
36A_2	—	—	3600.0
36A_3	—	—	3600.0
36A_4	—	—	3600.0
36A_5	—	—	3600.0
36A_6	—	—	3600.0
36A_7	—	—	3600.0
36A_8	—	—	3600.0
36A_9	—	—	3600.0
36A_10	—	—	3600.0
36A_11	—	—	3600.0
36A_12	—	—	3600.0
36A_13	—	—	3600.0
36A_14	—	—	3600.0
36A_15	—	—	3600.0
36A_16	—	—	3600.0
36A_17	—	—	3600.0
36A_18	—	—	3600.0
36A_19	—	—	3600.0
36A_20	—	—	3600.0
36A_21	—	—	3600.0
36A_22	—	—	3600.0
36A_23	—	—	3600.0
36A_24	—	—	3600.0
36A_25	—	—	3600.0
36A_26	—	—	3600.0
36A_27	—	—	3600.0
36A_28	—	—	3600.0
36A_29	—	—	3600.0
36A_30	—	—	3600.0
36A_31	—	—	3600.0
36A_32	—	—	3600.0
36A_33	—	—	3600.0
36A_34	—	—	3600.0
36A_35	—	—	3600.0
36A_36	—	—	3600.0
36A_37	—	—	3600.0
36A_38	—	—	3600.0
36A_39	—	—	3600.0
36A_40	—	—	3600.0
36A_41	—	—	3600.0
36A_42	—	—	3600.0
36A_43	—	—	3600.0
36A_44	—	—	3600.0
36A_45	—	—	3600.0
36A_46	—	—	3600.0
36A_47	—	—	3600.0
36A_48	—	—	3600.0
36A_49	—	—	3600.0
36A_50	—	—	3600.0

Table 4: Detailed results for the PA formulation.

Instance	Primal bound	Dual bound	Total time
24B_2	162	162	103.8
24B_3	164	164	194.1
24B_4	169	169	274.0
24B_5	181	178	3600.0
24B_6	211	207	3600.0
24B_7	187	187	3600.0
24B_8	161	161	195.4
24B_9	161	161	195.6
24B_10	165	165	264.6
24B_11	203	203	226.5
24B_12	193	193	181.6
24B_13	174	174	196.0
24B_14	189	189	212.6
24B_15	190	190	360.0
24B_16	213	213	810.9
24B_17	183	183	228.1
24B_18	166	166	299.3
24B_19	217	208	3600.0
24B_20	209	206	3600.0
24B_21	192	192	323.7
24B_22	208	206	3600.0
24B_23	182	182	587.9
24B_24	182	177	3600.0
24B_25	177	173	3600.0
24B_26	192	192	369.1
24B_27	208	187	3600.0
24B_28	210	207	3600.0
24B_29	192	192	579.1
24B_30	196	183	3600.0
24B_31	197	197	403.6
24B_32	183	183	319.9
24B_33	183	183	448.6
24B_34	234	188	3600.0
24B_35	200	199	3600.0
24B_36	173	173	264.1
24B_37	174	174	280.8
24B_38	196	196	852.0
24B_39	207	198	3600.0
24B_40	182	182	497.6
24B_41	199	199	849.5
24B_42	195	189	3600.0
24B_43	219	198	3600.0
24B_44	212	203	3600.0
24B_45	185	185	503.4
24B_46	163	163	331.5
24B_47	197	197	381.3
24B_48	181	181	581.3
24B_49	220	201	3600.0
24B_50	212	208	3600.0

Instance	Primal bound	Dual bound	Total time
36A_2	176	176	587.0
36A_3	201	166	3600.0
36A_4	205	147	3600.0
36A_5	215	137	3600.0
36A_6	210	172	3600.0
36A_7	280	82	3600.0
36A_8	234	120	3600.0
36A_9	327	110	3600.0
36A_10	299	76	3600.0
36A_11	264	117	3600.0
36A_12	224	123	3600.0
36A_13	307	102	3600.0
36A_14	331	192	3600.0
36A_15	277	81	3600.0
36A_16	325	74	3600.0
36A_17	262	110	3600.0
36A_18	265	128	3600.0
36A_19	333	173	3600.0
36A_20	330	71	3600.0
36A_21	270	101	3600.0
36A_22	229	128	3600.0
36A_23	261	95	3600.0
36A_24	269	99	3600.0
36A_25	215	139	3600.0
36A_26	225	129	3600.0
36A_27	225	126	3600.0
36A_28	254	125	3600.0
36A_29	221	125	3600.0
36A_30	248	150	3600.0
36A_31	219	126	3600.0
36A_32	238	125	3600.0
36A_33	196	196	1587.1
36A_34	178	178	887.1
36A_35	189	189	724.7
36A_36	197	197	2821.3
36A_37	232	122	3600.0
36A_38	239	129	3600.0
36A_39	209	173	3600.0
36A_40	234	130	3600.0
36A_41	196	180	3600.0
36A_42	193	169	3600.0
36A_43	199	174	3600.0
36A_44	194	181	3600.0
36A_45	193	193	1147.9
36A_46	185	185	1155.6
36A_47	192	192	953.1
36A_48	178	178	2834.9
36A_49	183	174	3600.0
36A_50	186	186	1172.8

Table 5: Detailed results for the GSPP formulation.

Instance	Primal bound	Dual bound	Total time
24B_2	162	162	0.0
24B_3	164	164	0.1
24B_4	169	169	0.1
24B_5	173	173	0.5
24B_6	189	189	0.8
24B_7	180	180	0.2
24B_8	161	161	0.1
24B_9	161	161	0.0
24B_10	165	165	0.1
24B_11	203	203	0.1
24B_12	193	193	0.1
24B_13	174	174	0.1
24B_14	176	176	0.1
24B_15	176	176	0.3
24B_16	199	199	0.5
24B_17	172	172	0.3
24B_18	166	166	0.1
24B_19	186	186	5.7
24B_20	200	200	0.9
24B_21	192	192	0.1
24B_22	186	184	3600.0
24B_23	169	169	96.0
24B_24	168	168	0.1
24B_25	161	161	0.2
24B_26	192	192	0.1
24B_27	183	183	0.8
24B_28	191	191	0.6
24B_29	171	171	148.3
24B_30	177	177	1.8
24B_31	197	197	0.1
24B_32	171	171	0.1
24B_33	183	183	0.1
24B_34	201	184	3600.0
24B_35	185	185	0.9
24B_36	173	173	0.1
24B_37	171	171	0.1
24B_38	182	182	0.3
24B_39	191	191	1.6
24B_40	176	176	0.8
24B_41	189	189	4.3
24B_42	175	175	1.7
24B_43	180	179	3600.0
24B_44	185	182	3600.0
24B_45	185	185	0.1
24B_46	162	162	0.1
24B_47	197	197	0.3
24B_48	181	181	0.1
24B_49	186	186	1.3
24B_50	197	197	0.1

Instance	Primal bound	Dual bound	Total time
36A_2	176	176	0.1
36A_3	173	173	146.4
36A_4	166	165	3600.0
36A_5	184	184	1.7
36A_6	176	172	3600.0
36A_7	190	174	3600.0
36A_8	179	178	3600.0
36A_9	197	197	1.9
36A_10	201	196	3600.0
36A_11	202	189	3600.0
36A_12	178	178	355.7
36A_13	183	180	3600.0
36A_14	208	191	3600.0
36A_15	189	168	3600.0
36A_16	196	173	3600.0
36A_17	198	197	3600.0
36A_18	194	194	1.7
36A_19	183	172	3600.0
36A_20	209	178	3600.0
36A_21	198	198	34.7
36A_22	180	175	3600.0
36A_23	191	184	3600.0
36A_24	198	198	4.2
36A_25	194	194	0.2
36A_26	173	173	174.9
36A_27	195	195	1.1
36A_28	187	187	63.1
36A_29	171	171	3.5
36A_30	191	191	0.2
36A_31	196	196	0.6
36A_32	180	180	115.9
36A_33	196	196	0.2
36A_34	178	178	0.1
36A_35	188	188	1.4
36A_36	197	197	0.1
36A_37	203	203	2.6
36A_38	193	193	13.0
36A_39	173	173	0.7
36A_40	183	177	3600.0
36A_41	180	180	0.7
36A_42	169	169	0.2
36A_43	174	174	0.4
36A_44	181	181	0.2
36A_45	193	193	0.2
36A_46	185	185	0.1
36A_47	192	192	0.2
36A_48	178	178	0.3
36A_49	174	174	0.5
36A_50	186	186	0.2

Table 6: Detailed results for the S formulation.

Instance	Primal bound	Dual bound	Total time
24B_2	162	162	0.3
24B_3	164	164	1.4
24B_4	169	169	13.1
24B_5	173	173	47.3
24B_6	189	189	339.2
24B_7	180	180	64.1
24B_8	161	161	1.4
24B_9	161	161	1.2
24B_10	165	165	8.4
24B_11	203	203	8.2
24B_12	193	193	5.4
24B_13	174	174	6.1
24B_14	176	176	12.8
24B_15	176	176	40.9
24B_16	199	199	77.5
24B_17	172	172	17.9
24B_18	166	166	18.2
24B_19	186	186	369.7
24B_20	200	200	120.7
24B_21	192	192	41.7
24B_22	186	183	3600.0
24B_23	169	169	97.2
24B_24	168	168	60.0
24B_25	161	161	29.8
24B_26	192	192	16.8
24B_27	183	183	123.3
24B_28	191	191	145.8
24B_29	171	171	127.1
24B_30	177	177	123.5
24B_31	197	197	20.6
24B_32	171	171	25.7
24B_33	183	183	22.4
24B_34	202	187	3600.0
24B_35	185	185	61.5
24B_36	173	173	8.2
24B_37	171	171	9.6
24B_38	182	182	60.8
24B_39	191	191	147.0
24B_40	176	176	16.0
24B_41	189	189	100.3
24B_42	175	175	166.4
24B_43	180	180	1776.5
24B_44	185	185	955.2
24B_45	185	185	49.6
24B_46	162	162	19.3
24B_47	197	197	65.8
24B_48	181	181	30.8
24B_49	186	186	172.5
24B_50	197	197	93.8

Instance	Primal bound	Dual bound	Total time
36A_2	176	176	25.3
36A_3	173	173	1056.3
36A_4	178	165	3600.0
36A_5	184	184	616.6
36A_6	185	172	3600.0
36A_7	199	175	3600.0
36A_8	195	178	3600.0
36A_9	197	197	654.3
36A_10	221	196	3600.0
36A_11	219	189	3600.0
36A_12	182	178	3600.0
36A_13	189	180	3600.0
36A_14	227	191	3600.0
36A_15	199	168	3600.0
36A_16	203	174	3600.0
36A_17	197	197	1985.3
36A_18	194	194	437.8
36A_19	185	172	3600.0
36A_20	217	178	3600.0
36A_21	201	198	3600.0
36A_22	183	175	3600.0
36A_23	207	184	3600.0
36A_24	198	198	635.6
36A_25	194	194	95.4
36A_26	182	173	3600.0
36A_27	195	195	1315.6
36A_28	187	187	968.1
36A_29	171	171	724.5
36A_30	191	191	141.8
36A_31	196	196	287.5
36A_32	181	180	3600.0
36A_33	196	196	163.2
36A_34	178	178	71.6
36A_35	188	188	38.6
36A_36	197	197	58.9
36A_37	203	203	404.1
36A_38	193	193	461.1
36A_39	173	173	198.4
36A_40	185	180	3600.0
36A_41	180	180	112.7
36A_42	169	169	83.6
36A_43	174	174	333.9
36A_44	181	181	64.0
36A_45	193	193	70.2
36A_46	185	185	41.1
36A_47	192	192	143.3
36A_48	178	178	115.7
36A_49	174	174	265.3
36A_50	186	186	132.4

Table 7: Detailed results for the TI formulation.