Časopis Pomorskog fakulteta Kotor – Journal of Maritime Sciences (JMS)
god. 27, br. 1/2026; Vol. 27, No. 1/2026

DOI: https://doi.org/10.56080/jms260503 UDC: 656.615:519.863
Original scientific paper

Computational Analysis of a Sedimentation Algorithm-Based Solver for Large-Scale Hybrid Berth Allocation Problem Instances

Stevan Kordić

Abstract: This paper examines the capability of a Sedimentation Algorithm–based hybrid solver to address the Hybrid Berth Allocation Problem with fixed handling times. The solver integrates a decomposition strategy, an efficient heuristic for generating upper bounds, and an exact optimization phase. A comprehensive computational study on 500 instances ranging from 5 to 60 vessels, in increments of 5 vessels, shows that the solver is both efficient and robust across a wide range of configurations. All instances with up to 50 vessels are solved within seconds, while only a small number of 55-vessel and 60-vessel test instances exhibit prolonged solving times. The results confirm that the hybrid structure effectively balances heuristic guidance and exact optimization, enabling the solver to handle even the most challenging test instances. The study also identifies promising directions for further improvement, particularly in strategies for determining vessel allocation sequences and in parallelized solving schemes.

Keywords: Berth Allocation Problem, Hybrid optimization, Exact methods, Decomposition, Sedimentation Algorithm, Hybrid solver.

1. Introduction

A container terminal (CT) can be viewed as an open system with two external access points through which container flows are processed. Decision-making at the port level has a direct impact on the operational performance of the terminal. Overall productivity depends on the efficient planning of vessel-related operations, including berth allocation, quay crane assignment, crane deployment, and container storage strategies. Among these tasks, berth allocation is a key problem: terminal managers must assign arriving vessels to available berths in a way that optimizes a predefined objective function and ensures smooth port operations.

Berth Allocation Problems (BAP) are commonly classified according to their spatial and temporal characteristics. This study focuses on the Hybrid Berth Allocation Problem (HBAP), in which the quay is divided into a finite

number of berths and each vessel may occupy multiple adjacent berths. The planning horizon is discretized into time units, enabling the evaluation of the objective function using integer arithmetic.

In computational complexity theory, problems in the class NP are those for which a proposed solution can be verified in polynomial time. A problem is classified as NP-hard if every problem in NP class can be reduced to it in polynomial time, making it at least as difficult as the most challenging NP problems. Since no polynomial-time algorithm is known for solving NP-hard problems, they typically require heuristic, metaheuristic, or specialized optimization techniques to obtain solutions within reasonable computational time. BAP has been shown to be NP hard [1], which makes it very complex to solve.

In this paper, we analyze the HBAP framework for large-scale scenarios, such as a two-week planning period with 13 berths. To assess the computational performance of large-scale instances of this problem, this study examines a hybrid optimization approach based on the Sedimentation Algorithm introduced in [2] and [3], which combines a decomposition stage, a heuristic procedure for generating upper bounds, and an exact optimization phase.

In the literature, exact approaches to the BAP, including the HBAP, are relatively scarce, as most studies rely on heuristic or meta-heuristic methods to obtain high-quality suboptimal solutions. According to the surveys by Bierwirth and Meisel in [4] and [5], exact methods account for only one quarter of published approaches, while the remaining three quarters are based on heuristic and meta-heuristic techniques.

Without aiming to provide a comprehensive overview of the literature related to the HBAP, we highlight several studies that address its solution approaches. The HBAP with fixed handling times has been examined by Chen and Hsieh [6] using a mixed integer programming formulation. Moorthy and Teo in [7] addressed the same problem by employing a precedence graph representation analyzed through the Project Evaluation and Review Technique. Dai et al. in [8] proposed a Simulated Annealing method for this variant of the problem, while Kovač [9] applied Bee Colony Optimization to the minimum cost version of the Hybrid Berth Allocation Problem with fixed handling times. Evolutionary algorithm was developed by El Hammouti et al. in [10]. Yang et al in [11] develop mixed-integer programming model with a buffer to address Dynamic HBAP.

The rest of this paper is organized as follows. Section 2 describes the HBAP model and the SEDA-based solver. Section 3 presents the

computational results. Finally, Section 4 provides the conclusions and outlines directions for future research.

2. HBAP Model and Solving Method

2.1. HBAP Model

The process of assigning vessels to berths is inherently complex and involves several interdependent decision problems, including the Berth Planning Problem, the Berth Allocation Problem, the Quay Crane Assignment Problem, and the Quay Crane Scheduling Problem. These components jointly determine the operational efficiency of a container terminal and must be coordinated to achieve optimal performance.

In this research, we focus specifically on the minimum-cost Hybrid Berth Allocation Problem. Our model is derived as a sub-model of the formulation presented by [12], from which we adopt the notation and retain only the berth-allocation component. The crane-assignment aspect is omitted, under the assumption that each berth is equipped with exactly one crane. In this paper, results for large-scale HBAP instances are obtained using a solver based on the Sedimentation Algorithm (SEDA). SEDA is a combinatorial algorithm that employs the input structure and objective function defined in the referenced sub-model. Further details on the combinatorial framework and the adaptation of SEDA for solving HBAP are provided in [2].

2.1.1. Assumptions

We adopt the following assumptions regarding the berthing of vessels in a container terminal:

Assumption 1. Each vessel has a pre-determined berthing time window. A penalty cost is incurred if the vessel berths earlier than scheduled, arrives late, or departs after its allocated time.

Assumption 2. Each vessel has a preferred berthing location. This preference may depend on factors such as the position of the storage area where its inbound and outbound containers are stacked, water depth, current strength and direction, or other operational considerations. A penalty applies if the vessel is assigned to a berth other than its preferred location.

Assumption 3. At any given time unit, only one container move (loading or unloading) can be performed at a berth. Furthermore, each berth is assumed to be equipped with exactly one crane dedicated to serving the vessel moored at that berth.

Assumption 3 is required to maintain compatibility with the model of [12], as our formulation addresses only the berth allocation component and does not incorporate crane assignment.

2.1.2. Input Data

Our model and algorithms use the following input data:

1. T: the total number of discrete time periods in the planning horizon.
2. m: the number of berths in the port.
3. l: the number of vessels considered within the planning horizon.
4. $vessel$: a sequence containing all relevant data for each vessel, defined as:

$$vessel = \{(ETA_k, a_k, b_k, d_k, s_k, C_{1k}, C_{2k}, C_{3k}, C_{4k}) \mid k = 1, \dots, l\}. \quad (1)$$

Each 9-tuple contains the following information for vessel k:

1. ETA_k: expected time of arrival of $vessel_k$.
2. a_k: processing time of $vessel_k$.
3. b_k: length of vessel k, expressed as the number of berths it occupies.
4. d_k: required departure time of $vessel_k$.
5. s_k: least-cost (preferred) berthing location of $vessel_k$.
6. C_{1k}: penalty cost if $vessel_k$ cannot berth at its preferred location.
7. C_{2k}: penalty cost per unit time for berthing before ETA_k.
8. C_{3k}: penalty cost per unit time for berthing after ETA_k.
9. C_{4k}: penalty cost per unit time for delayed departure after d_k.

In this paper, we present results for HBAP, where vessels may occupy between one and three adjacent berths. Accordingly, the value of the parameter b_k must lie within the range from 1 to 3 for all vessels.

2.1.3. Decision Variables and Domains

The formulation of BAP presented by [12] introduces the following decision variables. Although these variables are not essential for combinatorial algorithms, we include them here to provide a complete formal definition of the problem:

1. At_k: berthing time of $vessel_k$, $At_k \in \{1, \dots, T\}$.
2. Dt_k: departure time of $vessel_k$, $Dt_k \in \{1, \dots, T\}$.
3. Bi_k: lowest berth index allocated to $vessel_k$, $Bi_k \in \{1, \dots, m\}$.

4. X_{itk}: binary allocation variable which takes value 1 if berth i at the time t is allocated to $vessel_k$; otherwise, it takes the value 0. Obviously, $X_{itk} \in \{0,1\}$.

2.1.4. Constraints

A feasible solution to BAP must satisfy the following sets of constraints:

Constraint 1. At any time period t, each berth may be assigned to at most one vessel. This ensures that no two vessels occupy the same berth simultaneously.

Constraint 2. A berth may be allocated to a vessel only during the interval between its berthing and departure times, and only if the vessel fits within the available berth space.

2.1.5. Objective Function

We first introduce the auxiliary variable Z_k, which represents the sum of absolute distances between the preferred berthing location of *vessel*$_k$ and the berths allocated to it:

$$Z_k = \sum_{t=1}^{T} \sum_{i=1}^{m} \begin{cases} |i - s_k| & \text{if} \quad X_{itk} = 1, \\ 0 & \text{if} \quad X_{itk} = 0. \end{cases} \tag{2}$$

The penalty cost associated with $vessel_k$ is then calculated as:

$$\begin{aligned} VesselPen_k = C_{1k} Z_k + C_{2k}(ETA_k - At_k)^+ + C_{3k}(At_k - ETA_k)^+ \\ + C_{4k}(Dt_k - d_k)^+. \end{aligned} \tag{3}$$

where $(x)^+ = \max\{x, 0\}$.

The total port penalty cost, which serves as the objective function to be minimised, is given by:

$$VesselCost = \sum_{k=1}^{l} VesselPen_k. \tag{4}$$

This objective function minimises the combined cost of berth-position deviation, vessel waiting time, speed-up time, and tardiness. According to the classification proposed by [13], and based on the above formulation, our BAP instance can be categorised as:

disc | *stat* or *dyn* | *fix* | $\Sigma(w_1\ wait + w_2\ speed + w_3\ tard + w_4\ pos)$. (5)

2.2. HBAP Solver

The Sedimentation Algorithm (SEDA) is a general combinatorial optimisation method first introduced by the authors in [14] for solving BAP. In addition to BAP, SEDA has been successfully applied to other optimisation problems, such as *Max-SAT* [16] and the *Knapsack Problem*. SEDA belongs to

the class of *branch-and-bound* algorithms and employs a *backtracking* mechanism combined with *look-ahead* techniques to obtain exact solutions to optimisation problems. As previously noted, comprehensive details regarding the adaptation of SEDA for solving BAP are provided in [2].

To improve the solving times of the SEDA-based solver for BAP (and its special case, HBAP), two additional heuristics have been incorporated into the algorithm.

The first heuristic is based on a *Divide-and-Conquer* (D&C) strategy for solving both BAP and HBAP. It partitions the set of input vessels into subsets (sub-problems) and solves each subset using a BAP solver. The solutions of the sub-problems are then compared, and, if necessary, certain subsets are merged and solved again. This procedure is repeated until no further merging of subsets is justified. Full details of the *divide-and-conquer* implementation can be found in [3].

The second heuristic is the *Estimation & Rearrangement Heuristic* (ERH), which identifies a high-quality suboptimal solution to BAP (and HBAP) within a limited computation time. During this process, ERH determines an ordering of vessels in which the SEDA-based solver subsequently computes their optimal berthing positions and berthing times. This ordering is crucial for the efficiency of the SEDA-based solver.

The value of the objective function corresponding to the suboptimal solution produced by ERH is also of great importance. Since SEDA is a *branch-and-bound* algorithm, the search space can be significantly reduced when a good upper bound is provided by ERH. In some cases, ERH may even produce an optimal solution. When this occurs, two scenarios are possible. In the first scenario, ERH is able to verify that the obtained solution is optimal, and the search process terminates. In the second scenario, ERH finds an optimal solution but cannot prove its optimality within its limited computation time. In such cases, the final proof of optimality is delegated to the SEDA-based solver, which systematically explores the search space.

Based on the previously described solvers and heuristics, three solver configurations can be distinguished:

1. **SEDA** – a pure BAP solver based solely on the SEDA algorithm.
2. **ERH+SEDA** – a solver that first applies the ERH heuristic, which provides a good upper bound and an effective vessel ordering for SEDA.
3. **D&C+ERH+SEDA** – a solver that applies the *divide-and-conquer* strategy, where sub-problems are solved using the ERH+SEDA approach. Obviously, this hybrid solver combines a decomposition stage (D&C), a heuristic procedure for generating upper bounds and

effective vessel ordering (ERH), and an exact optimisation phase (SEDA).

Among these three configurations, **D&C+ERH+SEDA** is the most efficient, and all computational results presented in this paper are obtained using this solver. The full set of features and technical details of the above-mentioned algorithms can be found in the works of the authors previously cited.

3. Computational Results

3.1. Description of Test Instances

The experimental evaluation is conducted on test instances with 13 berths and a planning horizon of two weeks. The time horizon is discretised into 3-hour time units, resulting in a total of 112 time periods. The number of vessels ranges from 5 to 60, increasing in steps of 5. We denote this family of test instances as **HBAP 13×112**, referring to HBAP instances with 13 berths and 112 time units. When a third number is added, for example **HBAP 13×112×40**, it denotes the subset of instances with 13 berths, 112 time units, and 40 vessels. The set of test instances **HBAP 13×112** can be downloaded from [15].

Three types of vessels are included in the test population: feeder, medium, and mega. For each vessel type, Table 1 lists the corresponding share in the test population, the handling-time range, the penalty coefficients (expressed in units of US$ 1000), and the number of berths occupied in the case of HBAP.

Table 1 – *Test vessel specifications*

Vessel type	***Percentage of the test population***	***Handling time range in time units***	C_1	C_2	C_3	C_4	***Number of berths HBAP***
Feeder	*60%*	1 – 3	2	3	3	9	1
Medium	*30%*	4 – 5	3	6	6	18	2
Mega	*10%*	6 – 8	4	9	9	27	3

One example of a solved test instance, #73 from the **HBAP 13×112×60** set, is presented in Figure 1. The horizontal axis represents the time units, while the vertical axis corresponds to the berths. Vessels are shown as coloured rectangles with the vessel index displayed at the centre. Feeder vessels are depicted in green hues, medium vessels in yellow hues, and mega vessels in red hues.

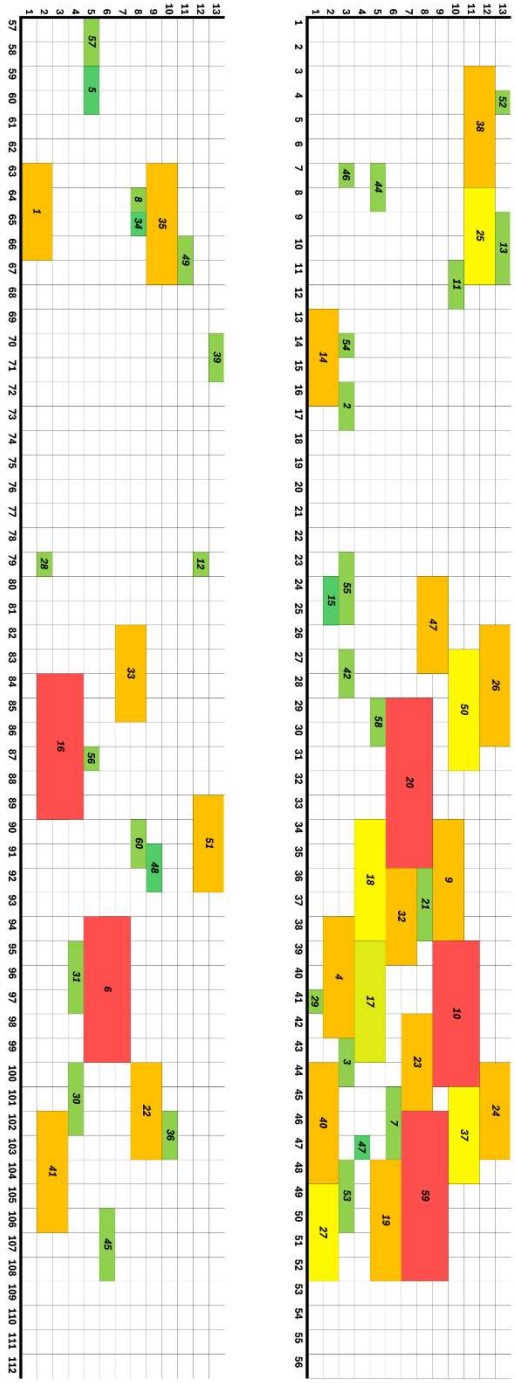

Fig. 1 – *Illustration of the solved test instance #73 from the HBAP 13×112×40 set. The horizontal axis represents the time units and is divided into two segments for improved visual clarity, while the vertical axis corresponds to the berths. Vessels are shown as coloured rectangles with their indices displayed at the centre. Feeder vessels are depicted in green hues, medium vessels in yellow hues, and mega vessels in red hues. The optimal solution for instance #73 has an objective value of 848, and the solver required 1397.796 s to obtain this solution.*

3.2. Experimental Setup

The **D&C+ERH+SEDA** solver was implemented in the *C programming language*. The code was compiled using the *Microsoft C/C++ Optimising Compiler* version 19.44.35228 for the x64 architecture. All computational tests were carried out on a machine equipped with an *AMD Ryzen 9 7940HS CPU* running at 4.00–5.20 GHz and 32 GB of RAM. The operating system used was *Microsoft Windows 11 Pro (64-bit)*.

3.3. Performance of D&C+ERH+SEDA Solver

The **D&C+ERH+SEDA** solver was tested on 500 instances of HBAP 13×112, with the number of vessels ranging from 5 to 60. For each instance set, the following performance indicators were recorded:

1. the minimum **D&C+ERH+SEDA** running time,
2. the median running time,
3. the average running time, and
4. the maximum running time.

The computational results for **HBAP 13×112** are presented in Table 2, and all running times are reported in seconds.

Table 2 – *Computational results for the HBAP 13×112*

l Number of vessels	Minimal	Median	Average	Maximal
5	0.000 s	**0.001 s**	0.001 s	0.005 s
10	0.001 s	**0.002 s**	0.002 s	0.048 s
15	0.002 s	**0.003 s**	0.005 s	0.159 s
20	0.002 s	**0.005 s**	0.023 s	0.436 s
25	0.004 s	**0.009 s**	0.069 s	0.824 s
30	0.004 s	**0.043 s**	0.171 s	1.588 s
35	0.006 s	**0.220 s**	0.339 s	2.732 s
40	0.007 s	**0.431 s**	0.685 s	65.846 s
45	0.012 s	**0.754 s**	1.094 s	21.698 s
50	0.021 s	**1.205 s**	3.436 s	188.001 s
55	0.041 s	**2.088 s**	15.698 s	1826.348 s
60	0.059 s	**3.193 s**	84.391 s	10610.448 s

The logarithmic-scale graph for **HBAP 13×112**, representing the solving times from Table 1, is shown in Figure 2.

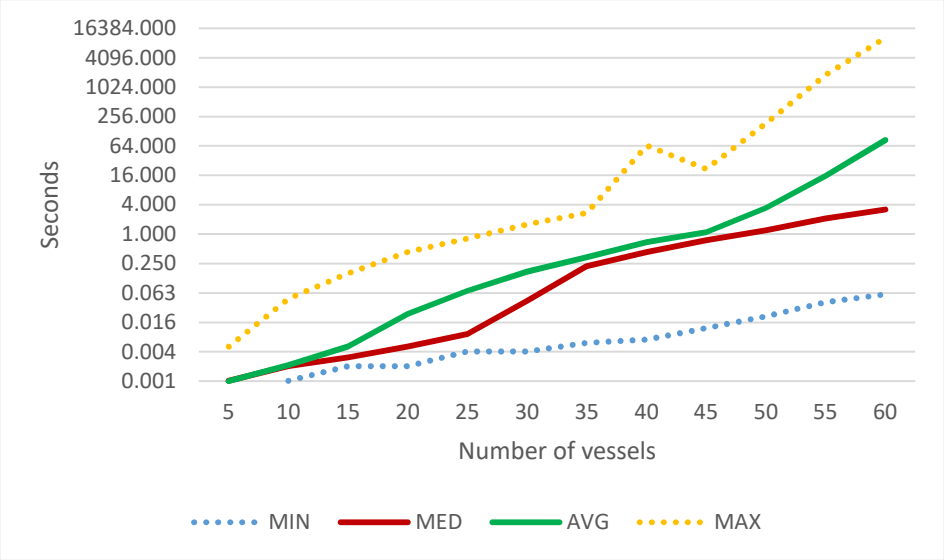

Fig. 2 – *Minimal, median, average, and maximal solving times of HBAP 13×112 instances for 5 to 60 vessels, in steps of 5*

Since the test instances with 55 and 60 vessels (i.e. **HBAP 13×112×55** and **HBAP 13×112×60**) are the most challenging to solve, these two sets will be examined in greater detail. The instances with fewer vessels are, as is evident from Table 1, easy for the **D&C+ERH+SEDA** solver, given that the maximal solving time is approximately 188 seconds, which is only slightly more than three minutes. Therefore, a detailed analysis of these cases is omitted.

Table 3 presents the solving-time distribution for the **HBAP 13×112×55** set of 500 test instances. The table reports the number and percentage of instances solved within right-closed time intervals $(a, b]$, where only the upper bound b is shown in the first column. All times are expressed in seconds. Figure 3 illustrates the solving-time distribution presented in Table 3.

Similarly to Table 3 and Fig. 3, which present the solving-time distribution for the **HBAP 13×112×55** test set, Table 4 and Figure 4 show the corresponding distributions for the **HBAP 13×112×60** set of instances.

Table 3 – *Solving-time distribution for HBAP 13×112×55*

t Upper bound	**Distribution**		**Cumulative Distribution**	
	Number of instances	*Percent of instances*	*Number of instances*	*Percent of instances*
0.25 s	23	4.60%	23	4.60%
0.5 s	39	7.80%	62	12.40%
0.75 s	23	4.60%	85	17.00%
1 s	31	6.20%	116	23.20%
2 s	119	23.80%	235	47.00%
4 s	159	31.80%	394	78.80%
8 s	58	11.60%	452	90.40%
15 s	19	3.80%	471	94.20%
30 s	6	1.20%	477	95.40%
60 s	2	0.40%	479	95.80%
120 s	4	0.80%	483	96.60%
240 s	9	1.80%	492	98.40%
480 s	5	1.00%	497	99.40%
960 s	2	0.40%	499	99.80%
1920 s	1	0.20%	500	100.00%
3840 s	0	0.00%	500	100.00%
12000 s	0	0.00%	500	100.00%

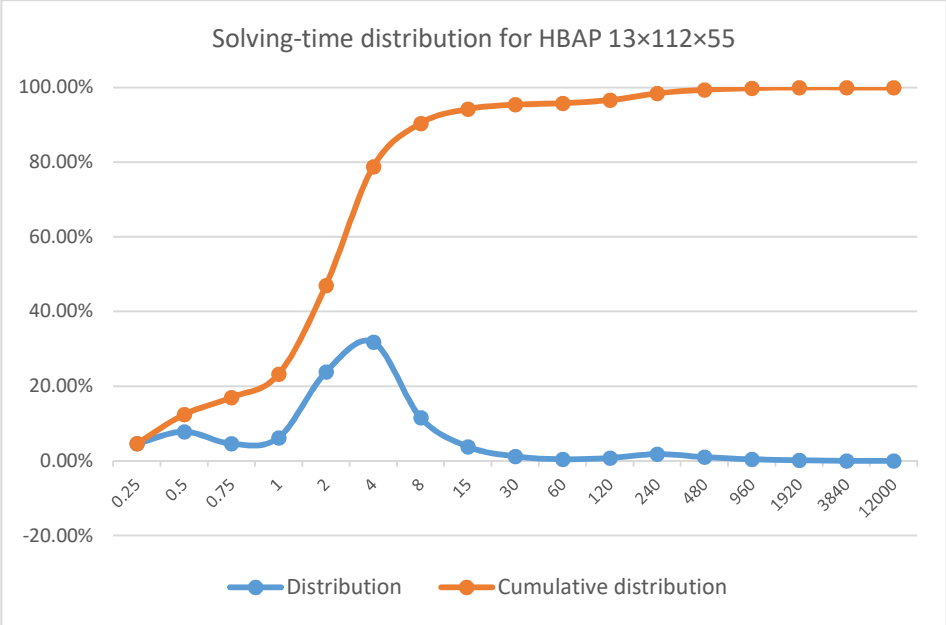

Fig. 3 – *Solving-time distribution for HBAP 13×112×55*

Table 4 – *Solving-time distribution for HBAP 13×112×60*

t Upper bound	**Distribution**		**Cumulative Distribution**	
	Number of instances	*Percent of instances*	*Number of instances*	*Percent of instances*
0.25 s	5	1.00%	5	1.00%
0.5 s	22	4.40%	27	5.40%
0.75 s	28	5.60%	55	11.00%
1 s	18	3.60%	73	14.60%
2 s	88	17.60%	161	32.20%
4 s	127	25.40%	288	57.60%
8 s	100	20.00%	388	77.60%
15 s	30	6.00%	418	83.60%
30 s	18	3.60%	436	87.20%
60 s	8	1.60%	444	88.80%
120 s	4	0.80%	448	89.60%
240 s	28	5.60%	476	95.20%
480 s	12	2.40%	488	97.60%
960 s	5	1.00%	493	98.60%
1920 s	2	0.40%	495	99.00%
3840 s	3	0.60%	498	99.60%
12000 s	2	0.40%	500	100.00%

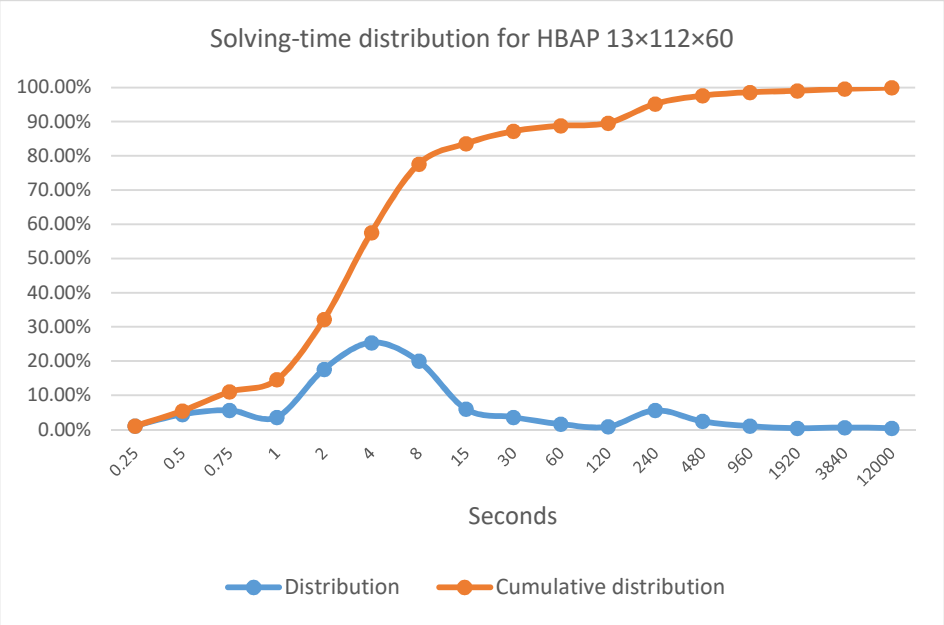

Fig. 4 – *Solving-time distribution for HBAP 13×112×60*

3.4. Analysis of Results

In Section 3.3, the performance of the **D&C+ERH+SEDA** solver was presented through tables and figures, accompanied only by the essential explanations required to interpret the reported data. In the following section, we provide a detailed analysis of these results, focusing on the behaviour of the solver across different instance sets, the distribution of solving times, and the factors that contribute to the observed performance patterns.

3.4.1. Solving-time Behaviour

From Table 2, which summarises the computational results for **HBAP 13×112**, it is evident that the **D&C+ERH+SEDA** solver is capable of handling this set of test instances efficiently. The overall maximal solving time is 10 610.448 seconds, slightly less than three hours. This value corresponds to the most extreme case, namely test instance #101 from the **HBAP 13×112×60** set with 60 vessels. All other instances are solved substantially faster.

All test instances with 5 to 50 vessels are solved very quickly. For example, the solving times for the **HBAP 13×112×50** set range from 0.021 to 188.001 seconds, with an average solving time of 3.436 seconds. It is important to note that the median solving time is only 1.205 seconds, which illustrates the high efficiency of the solver for instances with 55 vessels or fewer.

The most difficult sets of test instances are those with 55 and 60 vessels. For the **HBAP 13×112×55** set, the solving times range from 0.041 to 1826.348 seconds (slightly more than half an hour), with an average solving time of 15.698 seconds. As in the case of the sets with 50 vessels or fewer, the median solving time is much smaller than the average. For the 55-vessel set, the median solving time is only 2.088 seconds, which again demonstrates the efficiency of the **D&C+ERH+SEDA** solver. Moreover, Table 3 shows that 483 test instances (96.40%) are solved within 120 seconds, while only 17 instances (3.60%) require more than two minutes.

As expected, the most challenging set is **HBAP 13×112×60**, for which the solving times range from 0.059 to 10 610.448 seconds (slightly less than three hours), with an average solving time of 84.391 seconds. Again, as in all previous sets of test instances, the median solving time is 3.193 seconds and is much lower than the average, which confirms the strong overall efficiency of the **D&C+ERH+SEDA** solver. Table 4 shows that 476 test instances (95.20%) are solved within 240 seconds, while only 24 instances (4.80%) require more than two minutes. Only five instances are not solved within

1920 seconds (32 minutes), and among these, two require between two and three hours.

3.4.2. Hard Test Instances

While the majority of test instances are solved very quickly, a small subset exhibits significantly longer solving times and therefore requires special attention. These hard instances form the tail of the solving-time distribution and are crucial for understanding the limits of the D&C+ERH+SEDA solver. In this subsection, we examine the three most time-consuming test instances. Data related to their solving process are presented in Table 5.

Table 5 – *Data for the three most time-consuming test instances*

Test set	# *(Number of instance in the test set)*	ERH	Optimal solution	Gap	Solving time
HBAP 13×112×55	#173	891	885	0.68%	1826.348 s
HBAP 13×112×60	#101	975	963	1.25%	10610.448 s
HBAP 13×112×60	#200	1483	1475	0.54%	5901.373 s

Among the three most time-consuming test instances, one belongs to the **HBAP 13×112×55** set and the remaining two belong to the **HBAP 13×112×60** set. Instance #173 from the 55-vessel set requires 1 826.348 seconds to solve, which makes it the hardest case within that group. The other two hard instances, #101 and #200 from the 60-vessel set, require 10610.448 seconds and 5901.373 seconds, respectively. All three instances exhibit small but non-zero optimality gaps, ranging from 0.54% to 1.25%. This indicates that although the **ERH** phase provided good upper bounds for the **SEDA** solver, it did not produce a sufficiently effective ordering of vessels, which negatively affected the solving time. In addition, in all three cases the **D&C** strategy was unable to decompose the problem into smaller subproblems. After the initial partitioning, all subproblems quickly merged back into a single subproblem identical to the original instance with the full number of vessels (55 or 60), which rendered the **D&C** strategy ineffective. It should be noted that this behaviour appears in many other test instances as well, although in those cases it does not lead to such pronounced increases in solving time. These results show that although the solver is generally efficient, certain structural configurations in both the 55-vessel and 60-vessel sets can produce substantially more complex search spaces, leading to prolonged solving times.

3.4.3. Solver Robustness and Stability

The overall behaviour of the **D&C+ERH+SEDA** solver across all **HBAP 13×112** test instances indicates a high degree of robustness. Although solving times vary depending on the number of vessels and the structural characteristics of individual instances, the solver consistently produces optimal solutions. The stability of the solver is further supported by the performance of the ERH and D&C heuristics, which, despite occasional limitations, generally provide reliable guidance for the final SEDA phase.

When solving the **HBAP 13×112×55** set, **ERH** provided the exact solution for 481 instances (96.2%), while for the remaining 19 instances the maximal gap was 5.25% and the average non-zero gap was 1.56%. In the case of the **HBAP 13×112×60** set, **ERH** provided the exact solution for 442 instances (88.4%), while for the remaining 58 instances the maximal gap was 12.22% and the average non-zero gap was 1.71%. These indicators confirm the quality of the **ERH** upper bounds for the **SEDA** solver, although, as previously noted, ERH does not always provide the best vessel ordering for **SEDA**.

Since the density of vessels in the time–berth grid is very high in **HBAP 13×112** for the 55-vessel and 60-vessel sets, the D&C strategy was often unable to decompose the problem into smaller subproblems. In contrast, for the sets with 50 vessels or fewer, **D&C** was able to perform an effective decomposition, which significantly reduced solving times and thereby justifies the application of this strategy in solving HBAP instances.

3.4.4. Summary of Findings

The analysis of the computational results for **HBAP 13×112** demonstrates that the **D&C+ERH+SEDA** solver is highly efficient and robust across a wide range of test instances. Solving times remain low for all sets with up to 50 vessels, while the sets with 55 and 60 vessels introduce greater variability and a small number of hard cases. **ERH** provides strong upper bounds for the majority of instances, and although its vessel ordering is not always optimal, it reliably supports the final **SEDA** phase. The **D&C** strategy proves effective for less dense instances, while its limitations in the densest cases do not compromise the overall stability of the solver. Taken together, these findings confirm that the combined approach performs consistently well and is capable of handling even the most challenging **HBAP 13×112** instances.

4. Conclusion

As already demonstrated in Section 3.4.4, the proposed **D&C+ERH+SEDA** solver is capable of handling even the most challenging

HBAP 13×112 instances with up to 60 vessels, consistently producing optimal solutions and maintaining a high level of robustness across the entire test set.

The main advantage of the proposed approach lies in its hybrid structure, which combines heuristic guidance with exact optimisation. **ERH** efficiently generates high-quality upper bounds that substantially reduce the search space explored by **SEDA**, while **D&C** accelerates the solving process for less dense instances by decomposing the problem into smaller subproblems. This synergy enables the solver to maintain strong performance even in challenging configurations. In addition, the approach is conceptually simple, modular, and easily adaptable, making it suitable for integration into broader decision-support systems for berth allocation.

Future work may focus on improving the determination of vessel ordering for **SEDA**, since the quality of the ordering has a direct impact on solving times in the densest instances. Another promising direction is the exploration of parallelised solving strategies, in which multiple **SEDA** solvers, each using a different vessel ordering, would run concurrently. Such an approach would allow several orderings to be evaluated simultaneously, potentially offering a more effective alternative to searching for a single *"ideal"* ordering, whose structure is difficult to characterise and highly dependent on the specific instance.

Taken together, the results presented in this study demonstrate that the proposed hybrid approach offers a practical and effective framework for solving large-scale berth allocation problems. By combining decomposition, heuristic bounding, and exact optimisation within a unified structure, the solver achieves both efficiency and robustness across a wide spectrum of instances, including those of substantial size and complexity. These findings reinforce the value of hybrid optimisation strategies in maritime logistics and provide a solid foundation for further methodological advances and real-world applications.

References

[1] A. Lim, The berth planning problem, *Operations research letters*, 1998; 22(2): 105-110.

[2] S. Kordić, T. Davidović, N. Kovač, B. Dragović, Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, *Applied Mathematical Modelling*, 2016; 40(21-22): 8952-8973.

[3] S. Kordić, N. Kovač, T. Davidović, Divide and conquer approach to discrete berth allocation problem, *Scientific Bulletin" Mircea cel Batran" Naval Academy*. 2015; 18(2): 307.

[4] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, *European Journal of Operational Research*, 2010; 202: 615-627.

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

[6] J. H. Chen, D. H. Lee, J. X. Cao, A combinatorial benders' cuts algorithm for the quayside operation problem at container terminals, *Transportation Research Part E: Logistics and Transportation Review,* 2012; 48(1): 266-275.

[7] R. Moorthy, T. Chung-Piaw, Berth management in container terminal: The template design problem, *OR Spectrum*, 2006; 28(4): 495-518.

[8] J. Dai, W. Lin, M. Rajeeva, T. Chung-Piaw, Berth Allocation Planning Optimization in Container Terminals, in *Supply Chain Analysis. International Series In Operations Research & Mana*, Boston: Springer; 2008. p. 69-104.

[9] N. Kovač, Bee Colony Optimization Algorithm for the Minimum Cost Berth Allocation Problem, *XI Balkan Conference on Operational Research*, 2013; 245-254.

[10] I. El Hammouti, A. Lajjam, M. El Merouani, Y. Tabaa, An Evolutionary Algorithm Approach to Solve the Hybrid Berth Allocation Problem, in *Embedded Systems and Artificial Intelligence. Advances in Intelligent Systems and Computing;* 2020; Singapore: Springer.

[11] P. Yang, L. Cai, W. Guo, W. Li, A Proactive-Reactive Approach for Dynamic Hybrid Berth Allocation Problem Considering Vessels Arrival Delay, in *2023 IEEE Symposium Series on Computational Intelligence (SSCI);* 2023; Mexico City. p. 1753-1758.

[12] H. Rashidi, E. P. K. Tsang, Novel constraints satisfaction models for optimization problems in container terminals, *Applied Mathematical Modelling*, 2013; 37(6): 3601-3634.

[13] F. Meisel, C. Bierwirth, A framework for integrated berth allocation and crane operations planning in seaport container terminals, *Transportation Science*, 2013; 47(2): 131-147.

[14] S. Kordić, B. Dragović, D. Tatjana, N. Kovač, A Combinatorial Algorithm for Berth Allocation Problem in Container Port, in *Proceedings of International Association of Maritime Economists Conference;* 2012; Taipei. p. 1-15.

[15] S. Kordić, *ReaserchGate* [Online], 2026 [cited 2026 June 6.] Available from: https://www.researchgate.net/publication/406314433_Test_Instances_st01_for_Hybrid_Berth_Allocation_Problem_DBAP.

[16] S. Kordić, Application of Sedimentation Algorithm for Solving Max-SAT Problem, *Mathematica Montisnigri*, 2016; XXXVI: 45-57.

Submitted: 07/06/2026
Accepted: 21/06/2026

Stevan Kordić
University of Montenegro,
Faculty of Maritime Studies Kotor
Put I Bokeljske brigade 44,
85330 Kotor, Montenegro,
Email: stevan.kordic@ucg.ac.me