

Adaptive Metaheuristics for Multi-Objective Berth Allocation and Scheduling

Konstantinos Karathanasis ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
PIKEI New Technologies, Patras, Greece

Spyros Kontogiannis ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
Computer Technology Institute and Press “Diophantus”, Patras, Greece

Asterios Pegos ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
PIKEI New Technologies, Patras, Greece

Vasileios Sofianos ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece

Christos Zaroliagis ✉ 

Department of Computer Engineering and Informatics, University of Patras, Greece
Computer Technology Institute and Press “Diophantus”, Patras, Greece

Abstract

The *Berth Allocation and Scheduling Problem* (BASP) is a challenging combinatorial optimisation problem arising in container terminal operations, where decisions must balance conflicting goals including operational efficiency, economic cost, environmental impact, and infrastructure utilisation. This paper investigates the multi-objective discrete dynamic BASP with time windows and introduces a four-objective mixed-integer linear programming formulation that simultaneously captures vessel service efficiency, CO₂ emissions, operating cost, and berth workload balance.

To solve this computationally challenging problem, we develop and systematically evaluate adaptive multi-objective metaheuristic approaches. First, we adapt four evolutionary multi-objective algorithms through problem-specific variation operators and an adaptive operator selection mechanism. Second, we extend *Adaptive Large Neighbourhood Search* (ALNS) to the multi-objective BASP setting and propose two variants: SMOALNS, based on an external Pareto archive, and NSALNS, which integrates ALNS with NSGAII environmental selection.

Extensive computational experiments demonstrate that adaptive operator selection consistently improves evolutionary baselines and that the proposed multi-objective ALNS approaches achieve the best overall performance across benchmark instances. The results indicate that adaptive destroy–repair search mechanisms are particularly effective for highly constrained berth scheduling problems, providing high-quality approximations of the Pareto front for realistic port management scenarios.

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains; Applied computing → Multi-criterion optimization and decision-making

Keywords and phrases Berth Allocation and Scheduling, Multi-objective Optimization, Metaheuristics, Evolutionary Computation

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.7

Funding This work is partially supported by the EU I3 Instrument under GA No 101115116 (project AMBITIOUS) and by the University of Patras under GA No 83770 (programme “MEDICUS”).

1 Introduction

The growth of global maritime trade has placed increasing pressure on marine container terminals, making the efficient utilisation of berthing resources a critical operational challenge. Inefficient berth planning can cause vessel congestion, long waiting times, and increased



© Konstantinos Karathanasis, Spyros Kontogiannis, Asterios Pegos, Vasileios Sofianos, and Christos Zaroliagis;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 7; pp. 7:1–7:22



Open Access Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

operational and environmental costs. This challenge, commonly modelled as the *Berth Allocation and Scheduling Problem* (BASP), involves assigning arriving vessels to berthing positions and service times while satisfying spatial and temporal constraints. BASP variants may differ w.r.t. the quay layout: (i) *discrete* if the quay is divided into fixed berths, each serving at most one vessel at any time; (ii) *continuous*, if vessels can be assigned to any feasible position along the quay; and (iii) *hybrid*, if a vessel may occupy multiple adjacent berths, while each berth may simultaneously serve multiple vessels. They may also differ w.r.t. vessel arrival information: (i) *static*, if all vessels are already present awaiting to be served; (ii) *dynamic*, if vessels arrive over time at a priori known arrival times; and (iii) *uncertain*, if vessel arrival times are subject to variability. We denote these variants as XYBASP, where $X \in \{D, C, H\}$ represents the quay layout and $Y \in \{S, D, U\}$ the arrival-time setting.

Many practical BASP formulations also include *time-window constraints*, requiring vessels to be serviced within predefined intervals dictated by contractual, tidal, or operational considerations, which significantly increases problem complexity. Moreover, real-world berth planning involves multiple conflicting objectives, such as minimising vessel turnaround times, operational costs and environmental impact, while ensuring balanced berth utilisation.

Motivated by these challenges, we study the *multi-objective Discrete Dynamic Berth Allocation and Scheduling Problem with time windows* (moDDBASPTw). Our model simultaneously minimises total vessel service time in the system, CO₂ emissions and terminal operating costs, while promoting workload balance across berths. Given the intractability of the problem, we propose and experimentally evaluate several metaheuristic approaches that efficiently generate diverse high-quality approximations of the Pareto front, supporting informed decision-making in realistic port management environments.

Related Work. BASP has received considerable attention due to its impact on terminal efficiency and operational costs. The single-objective BASP is NP-hard [15], rendering exact approaches computationally prohibitive for large-scale instances and motivating the use of heuristic and metaheuristic methods. Several studies have focused on single-objective BASP variants. In [13], (single objective) DDBASPTw is formulated as a MILP inspired by vehicle routing models, and is approximately solved using constructive and improvement heuristics, as well as metaheuristics including ALNS, genetic algorithms, and cuckoo search.

The inherent limitations of single-objective formulations have led to increasing interest in multi-objective BASP, which not only inherits NP-hardness [2] but introduces additional complexity due to conflicting objectives. In [10], daytime preference is incorporated as an additional objective, and a bi-objective genetic algorithm is proposed to balance service efficiency and daytime workload. A Pareto-based evolutionary framework is introduced in [4] that jointly minimises makespan, waiting time, and deviation from priority schedules, incorporating problem-specific local search mechanisms. A multi-objective vessel scheduling framework accounting for cost efficiency, service reliability, and environmental impact is studied in [21]. Environmental aspects are explicitly addressed in [22] where shore-side electricity and emission reduction are integrated into berth allocation models, and in [17] where service efficiency and carbon emissions are jointly optimised using the enhanced multi-objective artificial bee colony (EMOABC) algorithm.

Despite these advances, none of the existing works has considered all the involved optimisation directions: (i) operational efficiency (e.g., expressed by total vessel waiting and service times), (ii) aggregate service cost, (iii) environmental footprint (e.g., expressed by CO₂ emissions) and (iv) balanced exploitation of the port infrastructure (e.g., as indicated by the workload distribution). Moreover, systematic comparisons of established multi-objective

metaheuristics within a common BASP setting are scarce, and algorithms with proven effectiveness in single-objective BASP, such as ALNS, have not yet been extensively explored in the multi-objective context. Recent studies have also extended the ALNS paradigm to multi-objective optimisation. Hybrid MOALNS frameworks, combining ALNS with population-based approaches such as NSGAII [6], SPEA2 [24], and MOEA/D [23], have been proposed for applications including transportation planning [18], green machine scheduling [5], disassembly line balancing [8], and vehicle routing [14]. These methods exploit adaptive destroy–repair operators, demonstrating the potential of MOALNS to effectively approximate Pareto fronts in complex combinatorial optimisation problems. Nevertheless, experimental evaluations are often restricted to small instances and a limited set of performance metrics.

Our Contribution. This paper makes the following contributions to moDDBASPTw.

1. We introduce, to the best of our knowledge, the first formulation that simultaneously considers objective functions in all the previously mentioned optimisation directions, namely, operational efficiency, total cost, environmental footprint, and balanced exploitation of the infrastructure.
2. We investigate the application of general-purpose multi-objective evolutionary algorithms (MOEAs) to moDDBASPTw by adapting four well-established approaches, namely NSGAII [6], NSGAIII [7], SPEA2 [24], and MOEA/D [23], through problem-specific variation operators. In particular, we design and implement five crossover operators and two mutation operators tailored to the structure of berth allocation schedules, and perform a systematic experimental analysis of their impact on MOEA performance. To the best of our knowledge, this constitutes the first systematic evaluation of crossover operators in the context of multi-objective BASP. We also implement the EMOABC [17] approach verbatim and include it in our experimental evaluation.
3. We adjust a well-established metaheuristic from the VRP literature, Adaptive Large Neighbourhood Search (ALNS) [19], to the moDDBASPTw setting. We propose two variants of ALNS: a simple multi-objective variant (SMOALNS), and a more sophisticated approach (NSALNS) that integrates ALNS within the selection and diversity-preservation mechanisms (non-dominated sorting) of NSGAII. Both these variants outperform all other approaches for solving moDDBASPTw, consistently achieving superior performance across nearly all evaluation metrics, thus demonstrating the effectiveness of combining large neighbourhood search with evolutionary multi-objective techniques.
4. We extend the adaptive operator selection mechanism originally introduced in ALNS to the considered MOEA frameworks. The resulting adaptive evolutionary variants dynamically select among multiple crossover and mutation operators during the search process. Extensive experimental evaluation demonstrates that adaptive operator selection improves upon deterministic operator configurations.

2 Problem Description & Mathematical Formulation

In this section, we introduce a *multi-objective* mathematical formulation of DDBASPTw, hereafter referred to as **moDDBASPTw**. The proposed formulation extends the *Mixed Integer Linear Programming* (MILP) formulation introduced in [13] for the (single-objective) DDBASPTw, inspired by its connection with the *Heterogeneous VRP*.

In particular, it considers four objective functions (see Figure 2 in the Appendix): (i) the minimisation of the *total time* each vessel spends within the port, reflecting operational efficiency; (ii) the minimisation of *CO₂ emissions* generated during a vessel’s stay in the

port, addressing environmental sustainability; (iii) the minimisation of the *overall berthing cost*, representing economic performance; and (iv) the minimisation of *load variation across berthing positions*, promoting a balanced and sustainable utilisation of port infrastructure. To discourage poor-quality schedules that simply deny service to vessels, the formulation explicitly accounts for penalties associated with vessels that are not assigned to any berth within their feasible time windows, thereby incentivising solutions that maximize service coverage while balancing the competing objectives. Additionally, our formulation explicitly defines appropriate constraints for the CO₂ emissions associated with each vessel-to-berth assignment. All other constraints remain the same as in [13]. Rather than seeking a single optimal solution, the objective is to identify a diverse set of approximate solutions that efficiently cover as much as possible the Pareto front, while satisfying all problem constraints.

Each vessel $i \in S$ is characterized by an estimated time of arrival T_i^{ea} and a requested time of departure T_i^{rd} (perceived as a soft deadline). The terminal enforces a delay policy T^{dp} , which specifies the maximum allowable delay beyond a vessel's requested departure time. This policy induces a hard time window $[T_i^{ea}, T_i^{rd} + T^{dp}]$, within which vessel i must be assigned to a berth and complete its service; otherwise, the vessel is denied service. Each vessel $i \in S$ carries a cargo quantity Q_i and has a desirable number of cranes that could ideally be allocated for its service. Furthermore, vessel i is associated with four penalty parameters P_i^t , P_i^e , P_i^c , and P_i^l , which represent the loss incurred by the terminal operator per objective function, in the event that the vessel is denied service.

Regarding the port infrastructure, the wharf is assumed to be partitioned into a set B of heterogeneous berths. Each berth $j \in B$ is characterized by its length L^j , draft D^j , per-hour operational cost C^j , and number N^j of operational homogeneous cranes (e.g., requiring the same amount of time K to load / unload one unit of cargo). Similarly, each vessel $i \in S$ is characterized by its length L_i and draft D_i , which must be compatible with the corresponding attributes of the berth to which it is assigned. Assigning vessel i to berth $j \in B$ results in different waiting times, handling times, departure delays and CO₂ emissions. The *waiting time* of i is defined as the elapsed time between its arrival and the commencement of its mooring at berth j . The *handling time* of i depends on the specific vessel–berth pairing and reflects the duration required by berth j to service vessel i , based on its cargo quantity and the operational characteristics of the berth. A *departure delay* for i is defined as the extra time (if any) from T_i^{rd} until the completion of its service at berth j . The *CO₂ emissions* of i depend on the specific vessel–berth pairing.

To formalize the problem, the following assumptions are adopted: (i) all the spatio-temporal details of the vessels arriving within the planning horizon are known in advance (discrete-dynamic scenario); (ii) each vessel requires a single, uninterrupted time interval for handling (no pre-emption of currently handled vessels); and (iii) all berths are empty at the beginning of the planning horizon. The proposed MILP formulation employs decision binary variables $x_{i,k}^j$, defined for all triples $(i, k, j) \in S^2 \times B$, which indicate whether vessel k is serviced immediately after vessel i in the servicing sequence of berth j . It further includes continuous temporal variables defined for pairs $(i, j) \in S \times B$, representing the mooring time t_i^j of vessel i at berth j (equal to T_i^{ea} if vessel i is not assigned to berth j), the handling time h_i of vessel i , the CO₂ emissions e_i of vessel i and the departure delay t_i^{ld} , defined as the time by which the completion of service exceeds the requested departure time T_i^{rd} . For more details about the formulation see Section A in the Appendix.

3 Solution Approaches

In multi-objective optimisation problems, the aim is to determine a maximal set of solutions that reflect different trade-offs among the competing objectives, so that every feasible solution is not superior across all objectives to (i.e., does not dominate) any solution within this set. Such a set is commonly referred to as *Pareto set*, or *Pareto front*. In this work, we approximately solve moDDBASPTw by employing custom-tailored variants of well-known multi-objective evolutionary algorithms (MOEAs) and two novel multi-objective variants of ALNS to construct large solution sets, which are not necessarily subsets of the Pareto front, but they nevertheless cover a large range of trade-offs among the competing objectives and none of the solutions is dominated any other solution within this set.

3.1 Baseline MOEA Approaches

The majority of MOEAs share a common algorithmic framework. They maintain a population of feasible solutions, initialized either randomly or heuristically. This population evolves over successive generations until a predefined termination criterion is fulfilled. In each generation, *mate selection* is used to choose individuals for reproduction, followed by a *variation (crossover/mutation) operator* to generate an offspring. Each offspring is then evaluated by computing all objective values, and *environmental selection* determines which individuals advance to the next generation. MOEAs are largely problem-independent, with each algorithm defining its own mate and environmental selection mechanisms. To adapt a MOEA to a specific problem, one needs only to specify how the population is initialized and how variation operators are applied to generate new solutions. In this study, we use four well-known MOEAs as our baseline algorithms: NSGAII [6], NSGAIII [7], SPEA2 [24], and MOEA/D [23]. All these baseline algorithms use the same population initialization procedure (cf. Section 3.2) and apply identical variation operators (cf. Section 3.3). They differ in the *mate selection* and *environmental selection* mechanisms, as well as in whether they return the final population or maintain an external archive of non-dominated solutions.

3.2 Population Initialization

The initial population is always generated using the constructive heuristic *aggregate pre-determined order Insertion* (agg-po-INS), which is a simple variant of the *predetermined order insertion* (po-INS) heuristic proposed in [13]. This heuristic incrementally constructs berth schedules by inserting one unassigned vessel at a time into partially built schedules. Based on a predetermined vessel ordering, at each step, all feasible insertion positions for the current vessel within each eligible berth are evaluated, and the position minimising the marginal cost increase is chosen, while preserving the relative order of previously assigned vessels. Mooring times of already scheduled vessels may be adjusted as long as temporal and operational constraints are satisfied. In our multi-objective setting, the marginal cost increase is computed as the increase in a weighted aggregation of the four objectives. For this operation to be meaningful, all objective values are normalized; this is consistently applied in this work whenever a weighted aggregation of the objectives is used.

To promote diversity of the initial population and to avoid bias towards any single objective, five predefined weight vectors are used: one assigning equal importance to all objectives (a weight of 0.25 each), and the other four emphasizing a particular objective (assigning to it a weight of 0.4) and equally splitting the remaining weight to the others (0.2 each). For a population of size N , approximately $N/5$ solutions are generated per weight vector. Diversity within each weight vector is achieved by varying the vessel insertion order, producing structurally distinct schedules and ensuring broad coverage of the objective space.

3.3 Variation Operators

Crossover Operators. A crossover operator generates an offspring solution by combining structural information from two parent solutions. In this work, we designed and evaluated five crossover operators, which are described below.

RandomParent (RP) processes vessels independently. For each vessel $i \in S$, if it was assigned to some berth by only one parent, the offspring inherits that assignment (berth and mooring time). If it was assigned to berths by both parents, one assignment is selected with equal probability. If the inherited assignment is infeasible w.r.t. assignments already copied into the offspring (e.g., temporal overlap on the same berth), it is discarded and vessel i is marked as unassigned. After all vessels are processed, unassigned vessels are reinserted using an **agg-po-INS**-based repair mechanism. The repair follows the **agg-po-INS** procedure described in Section 3.2, using equal objective weights (0.25 each), and operates on partially constructed berth schedules formed by the inherited assignments.

RandomBerth (RB) acts at berth level, rather than at vessel level. A berth $j \in B$ is selected uniformly at random, and its entire schedule in the first parent is replaced by its schedule in the second parent. As a consequence, vessels that were originally assigned to berth j in the first parent but are not present in the second parent's schedule of j become unassigned. Moreover, vessels copied from the second parent may have been assigned to other berths in the first parent; in such cases, their original assignments are removed to preserve uniqueness. All vessels that are left unassigned, are reinserted using the same **agg-po-INS**-based repair mechanism employed by the **randomParent** operator.

HalfBerths (HB) partitions the set of berths B in two disjoint subsets of equal size. The offspring inherits the schedules of berths in the first subset from the first parent and those of the remaining berths from the second parent, ignoring assignments that would duplicate vessels already inherited. Any unassigned vessels resulting from this process are reinserted using the **agg-po-INS**-based repair mechanism.

HalfTime (HT) performs recombination along the time dimension. Assignments from both parents are combined and ordered by the midpoint of their service intervals, $t_i^j + h_i/2$, and the temporal split $\bar{t}$ is defined as the median midpoint. The offspring inherits assignments from the first parent whose midpoints occur before $\bar{t}$ and from the second parent whose midpoints occur after $\bar{t}$, provided the vessel has not already been assigned. Assignments causing temporal conflicts, as well as vessels whose midpoints lie on the non-selected side of $\bar{t}$ in both parents, are marked as unassigned. All unassigned vessels are subsequently reinserted using the **agg-po-INS**-based repair mechanism.

Quarter (Q) combines **HalfBerths** and **HalfTime** by partitioning both the time horizon and the berth set. The temporal split $\bar{t}$ is defined as in **HalfTime**, while B is partitioned into two equal subsets B_1 and B_2 . For assignments with midpoints before $\bar{t}$, vessels assigned to berths in B_1 are inherited from the first parent and those in B_2 from the second parent; for assignments after $\bar{t}$, the inheritance is reversed. Duplicate assignments are ignored, and assignments causing temporal conflicts or omitted due to the combined partitioning are marked as unassigned. All unassigned vessels are finally reinserted using the **agg-po-INS**-based repair mechanism, yielding a feasible offspring.

Mutation Operators. The role of a mutation operator is to introduce controlled perturbations to existing solutions in order to enhance exploration while preserving feasibility. In this work, we design two mutation operators which are based on two well-known improvement heuristics from VRP literature, namely *ReInsert* and *Swap* [16]. In each case, before executing a mutation operator, the **agg-po-INS** repair mechanism is first applied to reinsert vessels that are not serviced in the current solution.

ReInsert is applied for a predefined number of iterations. In each iteration, a vessel is selected uniformly at random and removed from its assigned berth, allowing subsequently scheduled vessels to advance their mooring times if permitted by temporal constraints. The removed vessel is then reinserted using the **agg-po-INS** repair mechanism. The resulting solution is evaluated, and the changes are accepted only if the number of unserved vessels does not increase; otherwise, the solution is reverted. In the computational experiments, the number of iterations is set to 0.4 times the number of vessels.

Swap explores the neighbourhood through pairwise exchanges of vessel assignments between berths. A predefined number of vessel pairs is selected uniformly at random. For each pair, both vessels are first removed from their own berths and consequently they are reinserted into each other's original berths; if either vessel is incompatible with the new berth, the pair is skipped. All feasible insertion positions are evaluated, and the one minimising the marginal cost increase is selected according to the **agg-po-INS** principles, using equal objective weights. While reinsertion may shift the mooring times of other vessels, their relative order is preserved. Each swap is accepted only if it does not increase the number of unserved vessels; otherwise, the solution is reverted. In the computational experiments, the number of evaluated vessel pairs is set to 0.6 times the number of vessels.

3.4 Multi-Objective Adaptive Large Neighbourhood Search (MOALNS)

Adaptive Large Neighbourhood Search (**ALNS**) is a metaheuristic for single-objective optimisation problems that was introduced in [19] as an extension of Large Neighbourhood Search (**LNS**). It iteratively evolves the current solution by applying to it a pair of *destroy* and *repair* operators (exactly as **LNS**) but the selection process is now adaptive, choosing among multiple operators with probabilities proportional to their past performances. A custom-tailored variant of **ALNS** for **DDBASPt** was proposed in [13], where the experimental evaluation demonstrated its clear superiority over baseline (single-objective) evolutionary algorithms. Motivated by this observation, we propose two novel adaptations for **moDDBASPt**: the Simple Multi-Objective **ALNS** (**SMOALNS**) and the Non-dominated Sorting **ALNS** (**NSALNS**).

First, we introduce a generic **MOALNS** framework that defines the common structure shared by all proposed variants. At the beginning, an initial population of solutions is generated, as described in Section 3.2. Consequently, we initialise the weights of all the destroy and repair operators to equal values, implying uniformity in selection probabilities. Any additional data structures required by a specific variant are also initialized at this point. **MOALNS** proceeds iteratively until a predefined termination criterion is reached. In each generation, for each solution x in the current population $\mathcal{P}$, a destroy operator and a repair operator are selected according to their adaptive selection probabilities and applied consecutively, resulting in a new candidate solution x' . An acceptance function $\text{accept}(x', x, T)$ is then invoked, which is identical for all variants: If x' dominates x , then $\text{accept}(x', x, T)$ returns 1. If x dominates x' , it returns -1 . Otherwise (i.e., when neither x nor x' dominates the other), a simulated annealing-based criterion is applied, to allow the probabilistic acceptance of the newly generated solution even though it does not dominate the current solution, thereby promoting exploration. In particular, given the temperature parameter T , the value $e^{-diff/T}$ is computed, where $diff$ denotes the absolute difference between the values of x and x' w.r.t. the weighted sum of all the objectives. A random number r is drawn then uniformly from $[0, 1]$; if $r < e^{-diff/T}$, then $\text{accept}(x', x, T)$ returns 0 (to indicate the fact that a non-dominating solution was promoted this time), otherwise it returns -1 . After the acceptance decision is made, each variant applies its own procedure to handle the result. The scores of the selected operators are then updated based on the outcome of $\text{accept}(x', x, T)$.

These scores are subsequently used to update the operator weights. If $\text{accept}(x', x, T)$ returns 1, the selected operators receive the highest reward $rw_1 \in (0, 1)$; if it returns 0, they receive a smaller reward $rw_2 \in (0, rw_1)$; otherwise, no reward is granted to them. Once all solutions in the population have been processed, each variant performs its respective post generation operations.

A sequence of *updatePeriod* consecutive generations constitutes an *epoch*. At the end of the k -th epoch, the aggregate scores and weights (indicating new selection probabilities) for all the destroy and repair operators within the subsequent epoch $k + 1$ are calculated as follows: For each operator i ,

$$w_i^{k+1} := (1 - \rho) w_i^k + \rho \cdot \begin{cases} \frac{s_i^k}{u_i^k}, & \text{if } u_i^k > 0, \\ 0, & \text{otherwise,} \end{cases}, \quad p_i^{k+1} := \frac{w_i^{k+1}}{\sum_{j: \text{type}(j) = \text{type}(i)} w_j^{k+1}}, \quad (1)$$

where the type of each operator is from $\{D(\text{estroy}), R(\text{epair})\}$, w_i^k is the weight of operator i , s_i^k is its aggregate score, and u_i^k is the number of times it was selected, within epoch k . The new selection probability of i for epoch $k + 1$ is p_i^{k+1} , whereas $\rho \in (0, 1)$ determines the learning rate. This procedure allows MOALNS to adaptively favor operators that have been more successful in improving solutions, over time. At the same time, the temperature parameter is reduced from one epoch to the next, according to a geometric cooling schedule, i.e., $T^{k+1} := a \cdot T^k$, where $a \in (0, 1)$ is the cooling parameter.

Simple multi-objective ALNS (SMOALNS) and Non-Dominated Sorting ALNS (NSALNS).

SMOALNS maintains an external archive A that stores all mutually non-dominated solutions identified during the entire search. Starting with a population initialization step (cf. Section 3.2), for each new generation, if the acceptance function returns a value greater than or equal to zero (i.e., the current solution x is replaced by the new solution x'), then A is updated only if x' is not dominated by any solution already in A , and (in that case) all existing solutions in A which are dominated by x' are removed. SMOALNS does not apply any additional post generation population processing. Upon termination, the algorithm returns the external archive A obtained during the search, as an approximation of the Pareto front.

NSALNS incorporates the environmental selection principles of NSGAII. Instead of maintaining an external archive, NSALNS relies exclusively on population-based selection. During each generation, an offspring population $\mathcal{O}$ is constructed by appending each newly generated solution x' to $\mathcal{O}$, regardless of the outcome of $\text{accept}(x, x', T)$. After all solutions in $\mathcal{P}$ have been processed, non-dominated sorting (as in NSGAII) is performed on the combined population $\mathcal{P} \cup \mathcal{O}$, followed by crowding distance computation. The lexicographically best $|\mathcal{P}|$ solutions from $\mathcal{P} \cup \mathcal{O}$, w.r.t (smaller) dominance ranks and (larger) crowding distances, are selected to form the population for the next generation. Upon termination, NSALNS returns the final population $\mathcal{P}$ as the approximation of the Pareto front. Among the examined environmental selection mechanisms, the NSGAII strategy was retained due to its superior hypervolume and spread performance over all other baseline approaches in our experimental evaluation (cf. Section 4.1).

MOALNS Operators. Both variants employ the same set of destroy and repair operators.

Destroy Five destroy operators were implemented to selectively dismantle parts of the current solution: (i) random removals of assigned vessels, (ii) clearance of randomly selected berths, (iii) removal of vessels that significantly contribute to the total objective aggregated cost, (iv) removal of vessels that are spatially similar to a selected seed vessel

(i.e., vessels with similar length and draft) and (v) removal of vessels that are temporally similar to a selected seed vessel (i.e., vessels with close estimated time of arrival). The degree of destruction (*deg*) controls the proportion of the solution to be removed. A sufficiently high value of *deg* increases the likelihood of escaping from local optima.

Repair Six repair operators were implemented to reconstruct solutions following the destruction phase. Three of these operators are based on the **agg-po-INS** heuristic and differ only in the order in which the removed vessels are processed. Specifically, the vessel processing order is determined by (ascending) estimated-arrival times, or by (ascending) requested-departure times, or by (ascending) compatible-berths numbers. For all these repair operators, equal objective weights (0.25 per objective) are used in the aggregate cost computation. The fourth operator also relies on the **agg-po-INS** heuristic, processing vessels according to the order in which they were removed during the destruction phase, using again equal objective weights in the computation of the aggregate cost. The fifth operator is almost identical to the fourth, but employs randomly generated (instead of equal) objective weights. Finally, the sixth operator is based on a *k*-regret insertion strategy, which iteratively selects and inserts the vessel with the largest regret value from the set of removed and currently unassigned vessels. The *regret value* of a vessel is defined as the difference between the aggregate cost for its best feasible insertion position and the aggregate cost of its *k*-th best feasible insertion, using equal objective weights.

3.5 NSCSA with Decomposition and Adaptive Operator Selection

We developed also NSACSA/D, an enhanced version of NSCSA, a recently proposed metaheuristic [12] that combines the selection mechanisms of NSGAI with the exploration capabilities of the *Cuckoo Search Algorithm* (CSA) [9]. NSACSA/D adopts a decomposition-based framework and incorporates the adaptive operator selection mechanism of ALNS. The experimental results (cf. Section 4.1) revealed only a marginal performance improvement over the baseline methods. Therefore, we do not further analyze its behavior in detail.

3.6 Adaptive Operator Selection in Evolutionary and Genetic Algorithms

The performance of evolutionary and genetic algorithms is highly sensitive to the choice of crossover and mutation operators, and no single operator is expected to be universally optimal. In our case, the *a priori* confinement of the baseline algorithms to fixed combinations of variation operators may bias the search. On the other hand, it could be beneficial to equip each baseline MOEA with an adaptive operator selection mechanism to autonomously identify and exploit the most effective operators, while preserving exploration through probabilistic selection, similarly to the adaptive operator selection mechanism of ALNS.

To this end, adaptive variants of these algorithms are also introduced, denoted by adding an “A” to each original algorithm’s name: NSAGAI, NSAGAI, SPAEA2, and MOAEA/D. In particular, the adaptive mechanism for the destroy and repair operators in MOALNS (cf. Section 3.4) is adopted also in MOAEAs for selecting crossover and mutation operators. Each operator is associated with a weight, initialized uniformly and updated online based on observed performance. During reproduction, operators are chosen according to their adaptive selection probabilities (which are proportional to operator weights). After offspring evaluation, rewards are assigned independently for each parent: rw_1 if the offspring dominates the parent, rw_2 if they are mutually nondominated, and 0 if the parent dominates the offspring. For example, an offspring dominating both parents receives reward $rw_1 + rw_1$, while one dominating one parent and neither dominating nor being dominated by the other parent receives reward $rw_1 + rw_2$. Operator weights and the corresponding selection probabilities are updated periodically via exponential smoothing, similarly to MOALNS (cf. Equations (1)).

4 Experimental Evaluation

All experiments were conducted on a workstation running Ubuntu 22.04, equipped with an AMD Ryzen Threadripper 3960X processor (24 physical cores, 48 threads, base frequency of 3.80 GHz, and 128 MB L3 cache) and 256 GB of RAM. All algorithms were implemented in C++ and compiled using GCC with the `-O3` optimisation flag. The algorithm parameters were determined through a preliminary parameter tuning procedure, conducted on a representative subset of the instances used in the complete experimental evaluation, and were subsequently fixed for all experiments. For algorithms incorporating adaptive operator selection, the tuning process considered four parameters: the initial temperature T , cooling rate a , update period, and weight update parameter ρ . The candidate values investigated were: $T \in \{1, 2, 3, 4, 5, 6, 7\}$, $a \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $updatePeriod \in \{10, 15, 20\}$, and $\rho \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Parameter combinations were evaluated, and the configurations providing the best performance during the tuning phase were selected. The final parameter settings were: $T^{\text{MOALNS}} = T^{\text{MOEA}} = 3$, $a^{\text{MOALNS}} = a^{\text{MOEA}} = 0.3$, $\rho^{\text{MOALNS}} = 0.6$, $\rho^{\text{MOEA}} = 0.8$, and $updatePeriod = 10$ iterations for all methods. The operator rewards were set to $rw_1 = 1$ and $rw_2 = 0.4$. Finally, all MOALNS variants used a fixed destruction degree of $deg = 0.4$ in all experiments.

Performance Indicators. In our MILP formulation for 4-objective DDBASPTw, computing the (exact) Pareto front is computationally inefficient for all but toy instances. Even constructing a reliable approximation of the Pareto front through exact methods would be prohibitively expensive for meaningful problem sizes. Nevertheless, several performance indicators require a reference set that somehow approximates the Pareto front. Therefore, we adopt the following approach for the construction of a meaningful reference set, which is common practice in the literature [3, 20, 12, 17]: We first construct the union of the solution sets produced by all the algorithms under comparison across the experimental runs. From this set, only the non-dominated solutions are retained. The resulting set, called the *reference Pareto front*, is used as a surrogate for the unknown Pareto front when evaluating performance metrics that depend on it.

The experimental evaluation of each algorithm is done using the following set of commonly adopted performance indicators that capture computational efficiency, convergence, and diversity w.r.t. the reference Pareto front. **(i) Runtime (T):** It measures the total wall-clock time required by an algorithm to complete an optimisation run. It reflects computational efficiency and is particularly relevant for large-scale or time-sensitive applications. **(ii) Hypervolume (HV):** It is a unary performance indicator that simultaneously captures convergence and diversity. Given a solution set $S \subset \mathbb{R}^d$ and a single reference point $r \in \mathbb{R}^d$, the hypervolume of S is defined as the value of the Lebesgue measure $\Lambda(S)$ of the region that is weakly dominated by S and bounded above by r : $H(S) = \Lambda(\{q \in \mathbb{R}^d \mid \exists p \in S : p \leq q \leq r\})$. **(iii) Spread (Δ):** It evaluates the diversity and uniformity of the distribution of solutions returned by each algorithm. It measures how evenly the solutions cover the objective space and whether extreme regions of the front are adequately represented. A *lower* spread value indicates a more uniformly distributed solution set, whereas higher values suggest clustering or gaps along the front. This indicator is particularly useful when assessing an algorithm's ability to maintain solution diversity. **(iv) Inverted Generational Distance (IGD):** It assesses the quality of an obtained solution set S by measuring its average distance to a given reference Pareto front P^* . IGD computes the mean of distances of points in P^* from their nearest neighbour in S . Lower IGD values indicate better convergence and coverage of

the reference Pareto front, as they imply that the obtained solutions closely approximate the reference front across the entire objective space. In this work, the performance indicators are considered in lexicographic order of importance: $HV \prec \Delta \prec IGD \prec T$. We mainly focus on the quality of the obtained solution sets rather than execution time, as all algorithms have comparable running times (up to 15 minutes and differing by at most $3.6\times$).

4.1 Experimental Results

For the experimental evaluation, the time unit is set to 1 hour; for example, if $h_i = 2$, vessel i requires 2 hours to be handled by its assigned berth. The time for a single crane to load or discharge a unit of cargo is $K = 5$ minutes (1/12 hours), and the delay policy is $T^{dp} = 10$ hours. Vessel arrival times are considered within a single week (i.e., $\forall i \in S, T_i^{ea} \in [0, 168]$). Since no benchmark sets are available for the specific problem addressed in this study, all instances were generated using the data generator described in [13]. This generator exploits publicly available real-world data on vessel arrivals and berthing activity from Piraeus Container Terminal [1] (which can service up to eleven vessels simultaneously) for the creation of synthetic-data instances which reflect varying operational scales and at the same time are statistically comparable to real-world instances. Each benchmark data set is denoted as X-Y (X vessels, Y berths) and contains ten randomly generated instances of the given size; all tables report the arithmetic means across these random instances per data set. Columns include: “Algorithm” (method executed), “T” (average runtime in seconds), “HV” (hypervolume), “ Δ ” (spread), and “IGD” (inverted generational distance). The performance metrics are computed as follows. For methods returning populations, only non-dominated solutions are retained. The *reference Pareto front* is then constructed from the solution sets of all methods, and all points returned by the algorithms are normalized using the nadir and zenith points of this reference Pareto front. Metrics are subsequently calculated on the normalized solutions. For the HV metric, again following common practice, we consider the single reference point $r = (1.1, 1.1, 1.1, 1.1)$.

[EXP-I] Crossover Operator Comparison. Evolutionary algorithms are often sensitive to the choice of crossover. With multiple operators available in our framework, it is not immediately clear which provides a fair baseline configuration. To investigate this, each baseline algorithm was executed with all five crossover operators while keeping other parameters fixed. The **ReInsert** operator was used as the mutation operator in all cases, as preliminary experiments consistently indicated slightly superior performance over **Swap**. All MOEAs were run with a fixed population size of 100 solutions, and for 200 generations. For each crossover and instance, results were averaged across all baselines (i.e., the mean performance of a given crossover over all algorithms was reported). HV and Δ were used as performance metrics.

Table 1 reports the resulting average values for each problem instance. The results indicate that **RandomParent** (RP) consistently provides strong performance across instances, achieving the largest HV in all but two instances and the smallest Δ in all cases. Based on this observation, **RandomParent** is selected as the fixed crossover operator for all non-adaptive baseline algorithms in subsequent experiments. This choice ensures a fair and well-justified comparison with the adaptive variants, which are free to dynamically select among all available crossover operators.

[EXP-II] Baseline vs. Adaptive Variants. The comparison between baseline MOEAs and their adaptive counterparts are reported in Table 5 of Section B. Overall, the adaptive variants demonstrate slightly better solution quality and substantially faster execution times across most instances, reflecting the benefits of the **ALNS**-inspired adaptive operator selection.

■ **Table 1** Crossover averages (HV and Δ) across problem instances.

C/o	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$	HV $\uparrow$ $\Delta\downarrow$
	40 - 10	45 - 10	60 - 20	70 - 20	80 - 20	90 - 20
RP	1.022 0.733	1.119 0.701	1.058 0.781	0.996 0.713	1.103 0.771	1.12 0.684
RB	1.005 0.761	1.073 0.768	1.046 0.823	0.977 0.802	1.098 0.831	1.097 0.792
HB	0.981 0.831	1.035 0.812	1.026 0.849	0.97 0.799	1.084 0.833	1.073 0.795
HT	0.997 0.836	1.058 0.815	1.056 0.875	0.977 0.8	1.092 0.817	1.067 0.836
Q	0.995 0.801	1.069 0.806	1.049 0.851	0.967 0.785	1.094 0.82	1.068 0.783
C/o	100 - 25	120 - 30	135 - 30	140 - 35	150 - 35	
RP	1.055 0.733	1.094 0.725	1.058 0.696	1.091 0.722	1.087 0.748	
RB	1.05 0.831	1.09 0.806	1.068 0.784	1.065 0.829	1.072 0.871	
HB	1.023 0.806	1.072 0.812	1.034 0.754	1.063 0.788	1.101 0.845	
HT	1.035 0.798	1.082 0.784	1.049 0.751	1.077 0.798	1.099 0.835	
Q	1.032 0.792	1.073 0.759	1.036 0.743	1.062 0.775	1.057 0.797	

[EXP-III] Comparison with MOALNS Variants, NSACSA/D, and EMOABC. This experiment evaluates the performance of the proposed adaptive multi-objective evolutionary algorithms against NSACSA/D and an existing approach, namely EMOABC, as well as two MOALNS variants, namely SMOALNS and NSALNS. The EMOABC algorithm was selected as a baseline due to its recent application to multi-objective berth allocation problems [17]. It extends the conventional Artificial Bee Colony framework through several enhancement mechanisms, including a cooperative crossover operator, multiple neighbourhood search strategies, and archive-based local refinement. The implementation of EMOABC follows the procedure described in [17] as closely as possible to ensure a fair comparison. The original parameter settings were adopted, except for the parameter M , which was set to 10 to account for the higher number of objectives considered in this study. EMOABC, MOAEA/D, NSACSA/D, and SMOALNS return an external archive, whereas the remaining algorithms return a fixed-size set of solutions. We preserve this behavior to avoid altering the original design of the algorithms. All algorithms were evaluated using a population size of 100 individuals under two complementary experimental protocols. First, in the *Same Generations Experiment*, each algorithm was executed for 200 generations, enabling a direct comparison of solution quality under an identical number of evolutionary iterations. Second, in the *Time Limit Experiment*, all algorithms were compared under identical computational budgets to assess their performance w.r.t. computational efficiency.

[EXP-III.a] Same Generations Experiment. Table 2 summarizes the results obtained when all algorithms are executed for the same number of generations. Overall, the results indicate that the proposed MOALNS variants consistently outperform the competing approaches across all problem instances and performance indicators. Although the MOALNS variants generally require higher computational times than the adaptive baseline MOEAs, particularly for larger instances, they consistently achieve superior solution quality. This additional computational cost is mainly attributed to the higher complexity of the MOALNS repair operators compared to the crossover and mutation operators used in evolutionary algorithms. In particular, some repair mechanisms require repeated evaluation of candidate assignments within the solution reconstruction process, which increases the overall computational overhead.

Among the two variants, NSALNS demonstrates the most robust performance, achieving the highest HV values and the lowest Δ values across all tested instances. In contrast, SMOALNS obtains the lowest IGD values across all tested instances. Regarding NSACSA/D, the algorithm achieves higher HV values than all adaptive baseline variants and EMOABC in every instance. However, this advantage does not consistently hold also for the remaining indicators. Moreover, NSACSA/D exhibits substantially higher execution times than all other algorithms. Based on the adopted lexicographic ranking of performance indicators, NSALNS is

■ **Table 2** Same Generations Experiment.

Algorithm	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓
40-10					45-10				60-20			
NSAGAI	13.4	0.800	0.700	0.090	18.0	0.801	0.726	0.089	50.0	0.670	0.776	0.120
NSAGAIH	13.5	0.790	0.836	0.093	18.4	0.786	0.841	0.089	50.9	0.659	0.955	0.122
SPAEA2	13.8	0.782	0.734	0.122	18.6	0.793	0.723	0.108	50.1	0.662	0.832	0.136
MOAEA/D	14.1	0.799	0.764	0.085	19.3	0.791	0.745	0.080	50.5	0.644	0.676	0.126
EMOABC	7.5	0.710	0.683	0.137	10.7	0.636	0.792	0.166	43.6	0.616	0.762	0.155
NSACSA/D	27.1	0.816	0.783	0.085	35.5	0.815	0.821	0.084	101.9	0.709	0.781	0.108
SMOALNS	12.2	1.056	0.573	0.038	16.9	1.075	0.586	0.043	50.2	1.050	0.552	0.035
NSALNS	11.3	1.099	0.539	0.062	15.8	1.156	0.571	0.060	45.9	1.119	0.505	0.051
70-20					80-20				90-20			
NSAGAI	65.4	0.650	0.765	0.115	84.0	0.747	0.839	0.108	105.8	0.766	0.717	0.081
NSAGAIH	66.3	0.651	0.886	0.112	85.6	0.742	0.926	0.107	104.8	0.765	0.903	0.084
SPAEA2	66.0	0.646	0.842	0.125	85.5	0.745	0.894	0.118	106.5	0.762	0.863	0.089
MOAEA/D	66.0	0.629	0.693	0.112	85.3	0.735	0.692	0.129	101.4	0.738	0.784	0.085
EMOABC	69.9	0.578	0.703	0.154	95.0	0.708	0.777	0.132	136.7	0.631	0.718	0.151
NSACSA/D	135.5	0.670	0.785	0.102	178.9	0.779	0.845	0.093	214.2	0.769	0.828	0.076
SMOALNS	71.4	1.046	0.576	0.035	104.9	1.094	0.606	0.031	141.7	1.084	0.596	0.042
NSALNS	66.6	1.108	0.506	0.059	97.7	1.142	0.561	0.050	133.4	1.171	0.495	0.056
100-25					120-30				135-30			
NSAGAI	162.8	0.682	0.786	0.088	249.4	0.607	0.798	0.123	316.3	0.654	0.750	0.096
NSAGAIH	163.8	0.669	0.912	0.088	251.3	0.604	0.893	0.122	321.1	0.647	0.896	0.099
SPAEA2	161.8	0.679	0.897	0.098	248.6	0.598	0.905	0.133	314.7	0.654	0.845	0.109
MOAEA/D	158.9	0.667	0.729	0.083	247.2	0.585	0.723	0.120	305.3	0.619	0.721	0.098
EMOABC	186.8	0.630	0.768	0.121	299.6	0.548	0.740	0.157	394.9	0.598	0.680	0.143
NSACSA/D	333.9	0.691	0.788	0.081	523.1	0.614	0.789	0.111	656.9	0.684	0.861	0.091
SMOALNS	223.0	1.046	0.570	0.031	385.5	1.016	0.546	0.039	562.2	0.998	0.534	0.041
NSALNS	208.7	1.093	0.543	0.058	356.8	1.066	0.485	0.062	514.7	1.095	0.457	0.065
140-35					150-35							
NSAGAI	391.0	0.583	0.781	0.148	464.0	0.626	0.819	0.112				
NSAGAIH	397.7	0.571	0.932	0.155	467.0	0.620	0.945	0.115				
SPAEA2	390.5	0.580	0.871	0.157	464.8	0.624	0.918	0.117				
MOAEA/D	388.8	0.546	0.714	0.153	455.5	0.598	0.756	0.124				
EMOABC	473.7	0.558	0.776	0.172	547.5	0.580	0.727	0.151				
NSACSA/D	848.1	0.589	0.798	0.143	950.6	0.635	0.868	0.104				
SMOALNS	680.1	1.046	0.617	0.032	813.2	1.014	0.578	0.038				
NSALNS	636.0	1.083	0.567	0.055	740.4	1.062	0.526	0.062				

identified as the best-performing algorithm when considering the same number of generations.

[EXP-III.b] *Time Limit Experiment.* Since NSALNS requires higher computational effort than the baseline methods and EMOABC, this experiment assumes a fixed runtime budget. Specifically, the termination criterion is defined separately for each benchmark set according to the average computational time required by NSALNS to complete that benchmark set. It is then assessed whether the other algorithms, with lower computational requirements, could achieve competitive performance when provided with an equivalent computational budget. The results, reported in Table 3, confirm the trends observed in the previous experiments. The MOALNS variants continue to deliver the strongest overall performance across all test cases and evaluation metrics. Furthermore, NSACSA/D consistently outperforms the adaptive baseline MOEAs and EMOABC in terms of HV, although its performance w.r.t. the remaining indicators exhibits greater variability. Overall, these results further highlight the robustness and effectiveness of the MOALNS framework.

Two additional observations emerge from the experimental results. First, NSAGAI achieves higher HV values than NSAGAI in 21 out of 22 experiments presented in Tables 2 and 3. This is consistent with similar observations in the literature (e.g., [11]) indicating that, although NSAGAI is regarded as more suitable for problems with more than 3 objectives due to its environmental selection mechanism, it does not necessarily outperform NSAGAI. Second, the comparison between NSALNS and NSAGAI highlights the importance of the variation operators employed. Since the two algorithms differ only in their underlying

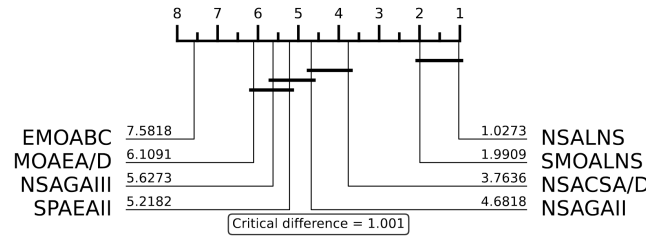
■ **Table 3** Time Limit Experiment.

Algorithm	T(s)	HV $\uparrow$	Δ $\downarrow$	IGD $\downarrow$	T(s)	HV $\uparrow$	Δ $\downarrow$	IGD $\downarrow$	T(s)	HV $\uparrow$	Δ $\downarrow$	IGD $\downarrow$	
40-10					45-10					60-20			
NSAGAI	12	0.776	0.662	0.081	16	0.765	0.767	0.081	45	0.645	0.790	0.117	
NSAGAIH		0.766	0.838	0.083		0.754	0.847	0.082		0.630	0.955	0.119	
SPAEA2		0.755	0.736	0.098		0.754	0.714	0.106		0.638	0.837	0.132	
MOAEA/D		0.756	0.748	0.084		0.753	0.706	0.071		0.614	0.680	0.122	
EMOABC		0.690	0.671	0.124		0.635	0.810	0.142		0.593	0.771	0.153	
NSACSA/D		0.781	0.794	0.077		0.776	0.815	0.076		0.676	0.761	0.101	
SMOALNS		1.012	0.586	0.056		0.990	0.602	0.065		1.006	0.589	0.055	
NSALNS		1.089	0.524	0.065		1.134	0.591	0.059		1.108	0.497	0.056	
70-20					80-20					90-20			
NSAGAI	67	0.687	0.766	0.100	98	0.751	0.804	0.088	134	0.795	0.737	0.077	
NSAGAIH		0.675	0.873	0.098		0.748	0.968	0.086		0.790	0.965	0.082	
SPAEA2		0.676	0.825	0.110		0.749	0.908	0.102		0.785	0.863	0.092	
MOAEA/D		0.660	0.691	0.094		0.735	0.702	0.099		0.760	0.736	0.081	
EMOABC		0.599	0.714	0.142		0.700	0.774	0.117		0.649	0.771	0.145	
NSACSA/D		0.708	0.779	0.086		0.772	0.859	0.082		0.798	0.835	0.079	
SMOALNS		0.988	0.624	0.057		1.063	0.629	0.041		1.050	0.586	0.062	
NSALNS		1.081	0.490	0.065		1.143	0.539	0.048		1.171	0.517	0.063	
100-25					120-30					135-30			
NSAGAI	209	0.678	0.732	0.073	357	0.637	0.784	0.093	515	0.682	0.739	0.066	
NSAGAIH		0.673	0.882	0.073		0.625	0.880	0.095		0.662	0.882	0.068	
SPAEA2		0.672	0.888	0.084		0.626	0.866	0.103		0.677	0.872	0.073	
MOAEA/D		0.668	0.726	0.068		0.623	0.731	0.085		0.650	0.744	0.061	
EMOABC		0.623	0.730	0.109		0.582	0.719	0.131		0.608	0.691	0.114	
NSACSA/D		0.694	0.790	0.067		0.647	0.781	0.084		0.703	0.894	0.062	
SMOALNS		0.972	0.641	0.044		0.989	0.606	0.052		0.965	0.600	0.056	
NSALNS		1.083	0.557	0.057		1.080	0.528	0.074		1.089	0.476	0.070	
140-35					150-35								
NSAGAI	636	0.644	0.775	0.087	741	0.656	0.782	0.076					
NSAGAIH		0.630	0.958	0.090		0.652	0.958	0.077					
SPAEA2		0.641	0.917	0.095		0.657	0.919	0.082					
MOAEA/D		0.623	0.727	0.084		0.635	0.724	0.076					
EMOABC		0.605	0.783	0.115		0.612	0.732	0.117					
NSACSA/D		0.656	0.806	0.084		0.677	0.875	0.074					
SMOALNS		1.031	0.669	0.039		0.970	0.648	0.048					
NSALNS		1.114	0.545	0.061		1.074	0.479	0.060					

search operators, the observed performance gap can be largely attributed to this component. The superior performance of NSALNS suggests that the destroy–repair paradigm is better suited to the moDDBASPTw than conventional crossover and mutation operators. A likely explanation is the highly constrained nature of berth allocation schedules, where problem-specific neighbourhood modifications can exploit structural information more effectively and preserve feasibility more reliably than recombination-based search. Appendix C presents the fraction of solutions contributed by each algorithm to the reference Pareto front.

[EXP-IV] Statistical Comparison. To assess whether the observed performance differences among the algorithms are statistically significant, a non-parametric Friedman test was conducted over the instances reported in Table 2, using HV as the performance metric. All algorithms were executed under the same number of generations, as in [EXP-III.a]. For each instance, algorithms were first ranked according to their HV values, and the Friedman test was then applied to these ranks to evaluate the null hypothesis that all methods perform equivalently. The Friedman test ($n = 110, k = 8$) rejected the null hypothesis ($\chi^2 = 601.4, p < 0.05$), indicating statistically significant differences among the compared algorithms.

To further identify which methods differ, we performed a post-hoc Nemenyi test and visualized the results using a critical difference (CD) diagram: the algorithms are ordered according to their average ranks across all instances. Methods connected by a horizontal line do not exhibit statistically significant differences at the 95% confidence level; methods not



■ **Figure 1** Critical difference diagram based on the Friedman test and Nemenyi post-hoc analysis for hypervolume performance across all benchmark instances.

connected by such a line are considered significantly different. Figure 1 shows that **NSALNS** achieves the best average rank (1.0273), followed closely by **SMOALNS** (1.9909); these two methods are not significantly different from each other, but both significantly outperform all remaining algorithms. **NSACSA/D** (3.7636) and **NSAGAI** (4.6818) form a second pairwise non-significant group. **NSAGAI** also overlaps with a third group comprising **NSAGAI** (5.6273) and **SPAEA2** (5.2182), and these two in turn overlap with a fourth group that includes **MOAEA/D** (6.1091), reflecting a chain of pairwise non-significant differences across the mid-ranked methods rather than a single homogeneous cluster. **EMOABC**, with the worst average rank (7.5818), is not connected to any other method and is therefore significantly outperformed by all other algorithms in the comparison. Overall, the CD analysis confirms that **NSALNS** and **SMOALNS** are the most consistently effective methods, with no statistically significant difference between them, while **NSACSA/D**, **NSAGAI**, **NSAGAI**, **SPAEA2**, and **MOAEA/D** form an overlapping mid-tier with no sharp boundaries between adjacent methods, and **EMOABC** performs significantly worse than every other algorithm tested. The same analysis for Δ and IGD confirmed the significant superiority of **MOALNS** variants over all other algorithms.

[EXP-V] Component ablation study for NSALNS. To investigate the contribution of individual destroy and repair operators within **NSALNS**, we conducted an ablation study in which the complete algorithm was compared against variants obtained by removing one operator at a time. For destroy operators, the results indicate that all operators contribute positively to the search process, with relatively small differences among them. The removal of the random removal operator resulted in the smallest performance degradation. This observation is expected, as random removal does not exploit (by design) any specific structural information from the current solution, unlike the more problem-oriented destroy operators. For repair operators, the impact was considerably more heterogeneous. While most repair operators resulted in moderate performance variations when removed, eliminating the k-regret insertion operator caused a substantial deterioration in solution quality. Furthermore, k-regret insertion is the most computationally expensive repair mechanism due to its intensive search procedure during solution reconstruction (cf. Section 3.4). To further analyze the role of k-regret insertion, we compared **NSALNS** with a variant using only this repair operator. Although k-regret insertion was identified as the most influential individual repair mechanism, the resulting algorithm was unable to outperform the complete **NSALNS**. This indicates that the effectiveness of **NSALNS** arises from the complementary interaction among multiple repair strategies rather than from a single dominant operator. Finally, we evaluated reduced variants containing all destroy operators and different subsets of repair operators, specifically all possible pairs of repair operators that include the k-regret insertion operator. The results showed that combining k-regret insertion with the random-weight agg-po-INS-based repair operator achieves performance close to the full **NSALNS**. Therefore,

the results suggest that k-regret insertion and the random-weight **agg-po-INS**-based repair operator provide particularly valuable contributions to the search process; however, the ablation results also demonstrate that the overall effectiveness of the algorithm relies on the adaptive combination of multiple operators rather than on any individual repair strategy.

5 Conclusions

In this work, we proposed a new multi-objective formulation for **moDDBASPt_w**, consisting of four objectives that capture realistic aspects and operational variables of container terminals. To solve this optimisation model, we designed multiple variation operators and employed well-established MOEAs, which we enhanced with adaptive operator selection mechanisms. Additionally, we introduced an enhanced version of **NSCSA**, named **NSACSA/D**, and we adapted **ALNS** to the multi-objective setting, providing **SMOALNS** and **NSALNS**. Our experimental results highlighted the advantage of incorporating adaptive mechanisms into evolutionary algorithms and demonstrated the superior performance of **SMOALNS** and **NSALNS**, which consistently outperformed all other algorithms. These findings provide an interesting insight: in complex multi-objective problems, such as **moDDBASPt_w**, generating new solutions via destroy-and-repair operators can lead to higher-quality solutions and faster convergence than relying solely on classical variation mechanisms. Future work will investigate different adaptivity mechanisms for operator selection and will explore their impact on convergence speed, solution quality, and stability in dynamic and large-scale problem instances.

References

- 1 Piraeus Container Terminal S. A. Piraeus container terminal. URL: www.pct.com.gr.
- 2 Tobias Buer and Herbert Kopfer. A pareto-metaheuristic for a bi-objective winner determination problem in a combinatorial reverse auction. *Computers & Operations Research*, 41:208–220, 2014. doi:10.1016/j.cor.2013.04.004.
- 3 Kamal Chakkarapani, Thyagarajan Thangavelu, Kalpana Dharmalingam, and Pragadheeshwaran Thandavarayan. Multiobjective design optimization and analysis of magnetic flux distribution for slotless permanent magnet brushless dc motor using evolutionary algorithms. *Journal of Magnetism and Magnetic Materials*, 476:524–537, 2019. doi:10.1016/j.jmmm.2019.01.029.
- 4 C. Y. Cheong, K. C. Tan, D. K. Liu, and C. J. Lin. Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research*, 180(1):63–103, November 2010. doi:10.1007/S10479-008-0493-0.
- 5 Luciano P. Cota, Frederico G. Guimarães, Roberto G. Ribeiro, Ivan R. Meneghini, Fernando B. de Oliveira, Marcone J.F. Souza, and Patrick Siarry. An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem. *Swarm and Evolutionary Computation*, 51:100601, 2019. doi:10.1016/j.swevo.2019.100601.
- 6 Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6(2):182–197, 2002. doi:10.1109/4235.996017.
- 7 Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans. Evol. Comput.*, 18(4):577–601, 2014. doi:10.1109/TEVC.2013.2281535.
- 8 Amir M. Fathollahi-Fard, Peng Wu, Guangdong Tian, Dexin Yu, Tongzhu Zhang, Jianwei Yang, and Kuan Yew Wong. An efficient multi-objective adaptive large neighborhood search algorithm for solving a disassembly line balancing model considering idle rate, smoothness, labor cost, and energy consumption. *Expert Systems with Applications*, 250:123908, 2024. doi:10.1016/j.eswa.2024.123908.

- 9 Amir Hossein Gandomi, Xin-She Yang, and Amir Hossein Alavi. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Eng. Comput.*, 29(1):17–35, 2013. doi:10.1007/S00366-011-0241-Y.
- 10 Zhi-Hua Hu. Multi-objective genetic algorithm for berth allocation problem considering daytime preference. *Computers & Industrial Engineering*, 89:2–14, 2015. Maritime logistics and transportation intelligence. doi:10.1016/j.cie.2015.04.035.
- 11 Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3045–3052. IEEE, 2016. doi:10.1109/CEC.2016.7744174.
- 12 Konstantinos Karathanasis, Spyros Kontogiannis, and Christos Zaroliagis. Task orchestration in the cloud continuum via multi-objective evolutionary algorithms. In *International Symposium on Algorithmic Aspects of Cloud Computing*, pages 73–93. Springer, 2025. doi:10.1007/978-3-032-13744-9_6.
- 13 Konstantinos Karathanasis, Spyros C. Kontogiannis, Asterios Pegos, Vasileios Sofianos, and Christos D. Zaroliagis. Vrp-inspired techniques for discrete dynamic berth allocation and scheduling. In *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 137 of *OASICS*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/OASICS.ATMOS.2025.6.
- 14 Abdelfettah LABDIAD. Empirical study of mopso, nsga ii, aco and alns comparison in multiobjective vehicle routing problem with flexible time windows. *IAENG International Journal of Computer Science*, 52(10), 2025.
- 15 Andrew Lim. The berth planning problem. *Operations Research Letters*, 22(2):105–110, 1998. doi:10.1016/S0167-6377(98)00010-8.
- 16 Fei Liu, Chengyu Lu, Lin Gui, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. Heuristics for vehicle routing problem: A survey and recent advances. *CoRR*, abs/2303.04147, 2023. doi:10.48550/arXiv.2303.04147.
- 17 Xiaomeng Ma and Xujin Pu. Low-carbon berth allocation: An analysis of the effectiveness of an enhanced multi-objective artificial bee colony algorithm based on a case study. *Ocean & Coastal Management*, 261:107529, 2025. doi:10.1016/j.ocecoaman.2024.107529.
- 18 Rui Qi, Jun qing Li, and Xiao feng Liu. A knowledge-driven multiobjective optimization algorithm for the transportation of assembled prefabricated components with multi-frequency visits. *Automation in Construction*, 152:104944, 2023. doi:10.1016/j.autcon.2023.104944.
- 19 Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. doi:10.1287/trsc.1050.0135.
- 20 Zhongshi Shao, Dechang Pi, and Weishi Shao. A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. *Knowledge-Based Systems*, 165:110–131, 2019. doi:10.1016/j.knosys.2018.11.021.
- 21 Xin Wen, Qiong Chen, Yu-Qi Yin, Yui-yip Lau, and Maxim A. Dulebenets. Multi-objective optimization for ship scheduling with port congestion and environmental considerations. *Journal of Marine Science and Engineering*, 12(1), 2024. doi:10.3390/jmse12010114.
- 22 Mingyuan Yue, Yubing Wang, Siqing Guo, Lei Dai, and Hao Hu. A multi-objective optimization study of berth scheduling considering shore side electricity supply. *Ocean & Coastal Management*, 261:107500, 2025. doi:10.1016/j.ocecoaman.2024.107500.
- 23 Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.*, 11(6):712–731, 2007. doi:10.1109/TEVC.2007.892759.
- 24 Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK report*, 103, 2001.

A

 Notation & Formulation

This section presents the mathematical formulations of the objectives and briefly describes the constraints. For further details, see the single-objective formulation in [13]. The first objective function (2) aims to minimise the total waiting time, port stay duration, and departure delay of each vessel within the port. The second objective function (3) seeks to minimise both the fuel consumption and the CO₂ emissions generated both during a vessel's entry to and exit from the port and throughout its stay within the port. The third objective function (4) focuses on minimising the total operational cost incurred by the port for the berthing and servicing of a vessel. The fourth objective function (5)¹ aims to minimise load variation across berthing positions, that is, to distribute the workload evenly among berths, thereby reducing potential equipment wear and, consequently, limiting additional maintenance costs incurred by the port.

■ **Table 4** Notation. **Boldfaced** symbols indicate decision variables.

Symbol	Description
$i, k \in S, j \in B$	Indices as subscript for vessels from S and as superscript for berths from B .
$x_{i,k}^j$	1 if $i, k \in S$ are both assigned to berth $j \in B$ for their service, with k being served right after i , 0.0 otherwise.
t_i^j	Mooring time of $i \in S$ at berth $j \in B$ to start being served.
t_i^{ld}	Late departure time for $i \in S$ from its assigned berth.
e_i	CO ₂ Emission Factor for $i \in S$ at its assigned berth.
T_i^{ea}, T_i^{rd}	Estimated Time of Arrival and Requested Time of Departure for $i \in S$.
H_i^j, H_i^-	Handling time for $i \in S$ at berth $j \in B$, Worst Handling time for $i \in S$ across all berths.
$F_i^{wt}, F_i^{ht}, F_i^{SC}$	Fuel Consumption rate while waiting to moore and Fuel Consumption rate while vessel being serviced for $i \in S$. Specific Fuel Consumption for auxiliary engines (tn/kWh).
N_i, N^j	Number of cranes requested by vessel $i \in S$. Number of cranes operating at berth $j \in B$.
K	Time of a single crane to load/discharge a unit quantity.
Q_i, Q_i^f	Cargo Quantity of vessel $i \in S$, average load factor of the containers on vessel $i \in S$.
$P_i^t, P_i^e, P_i^c, P_i^l$	Penalties for not servicing $i \in S$ for each objective respectively.
L_i, D_i	Length and draft of vessel $i \in S$, respectively.
L^j, D^j	Length and draft of berth $j \in B$, respectively.
C^j, C^-	Operational cost for $j \in B$ and worst operational cost across B .
T^{dp}, A	Delay Policy (time units), Fuel Consumption Coefficient
p^{CO_2}, h^{CO_2}	CO ₂ /hour cost generated by vessel operation in port and CO ₂ cost for serving one TEU.
Θ_b, Θ_c	Berth reservation fee(€/m/hour) and crane operation fee (€/hour).

The model is subject to a standard set of spatio-temporal constraints. Each vessel may be assigned to at most one berth. Spatial feasibility is enforced through berth length and draft compatibility. The handling time of vessel i depends on its cargo quantity Q_i , the time K required for a single crane to handle one unit of cargo, the number N^j of operational cranes available at berth j , and the maximum number of cranes that vessel i can effectively utilise. The actual number of cranes employed during service is given by the minimum of the berth's available cranes and the vessel's crane requirement. Accordingly, the handling time of vessel i at berth j is proportional to its cargo quantity divided by this effective number of cranes, scaled by K . This value is precomputed for each vessel–berth pair (i, j) and stored as the parameter H_i^j , obtained by applying a ceiling operation to preserve data integrity.

¹ Although 5 is presented as nonlinear formulation, it can be linearized using McCormick envelopes.

$$\min \sum_{i \in S} \left[\sum_{j \in B} \left(t_i^j - T_i^{ea} \right) + (t_i^{ld} + h_i) + P_i^t \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (2)$$

$$\min \sum_{i \in S} \left[\sum_{j \in B} \left(F_i^{wt} \cdot (t_i^j - T_i^{ea}) \right) + (F_i^{ht} \cdot h_i + e_i) + P_i^e \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (3)$$

$$\min \sum_{i \in S} \left[\sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} (x_{i,k}^j \cdot h_i \cdot C^j) + P_i^c \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (4)$$

$$\min \sum_{j \in B} \left[\sum_{i \in S} \sum_{k \in S \cup \{d\}: k \neq i} \left(H_i^j \cdot x_{i,k}^j - \frac{1}{|B|} \sum_{j' \in B} H_i^{j'} \cdot x_{i,k}^{j'} \right) \right]^2 + \sum_{i \in S} \left[P^l \cdot \left(1 - \sum_{j \in B} \sum_{k \in S \cup \{d\}: k \neq i} x_{i,k}^j \right) \right] \quad (5)$$

■ **Figure 2** Objective functions for moDDBASptw. Parameters in **bold** indicate decision, or temporal variables. Parameters in capital letters indicate constants, which are provided as input.

The CO₂ emissions of vessel i , depend on its handling time H_i^j , its cargo quantity Q_i , the hourly CO₂ cost p^{CO_2} generated by vessel operation across the whole port and CO₂ cost h^{CO_2} for servicing. These values are again precomputed and stored. A hard time-window restriction is imposed, ensuring that any vessel i that is granted service is handled within the interval $[T_i^{ea}, T_i^{rd} + T^{dp}]$. The simultaneous servicing of two different vessels at the same berth is prohibited. Finally, the departure delay t_i^{ld} is defined as the maximum between zero and the difference between the actual departure time and T_i^{rd} .

The penalties for all objectives are designed so that the algorithm is never incentivized to leave a vessel unassigned. The time-related penalty $P_i^t = (T_i^{rd} + T^{dp} - T_i^{ea}) + T^{dp} + \max_j H_i^j = T_i^{rd} + 2T^{dp} - T_i^{ea} + H_i^-$, provides a conservative upper bound on the total time vessel i may spend in the port if it cannot be serviced. It is obtained by summing the maximum possible waiting time, the maximum allowable departure delay, and the maximum handling time. Here, $H_i^- = \max_{j \in B} H_i^j$ denotes the longest handling time of vessel i across all berths. Similarly, the energy-related penalty $P_i^e = (T_i^{rd} + T^{dp} - T_i^{ea}) F_i^{wt} + H_i^- F_i^{ht} + E_i^-$, defines a conservative upper bound on the maximum CO₂ emissions that vessel i could generate if left unassigned. This penalty consists of three components. The first term captures emissions during waiting, computed as the product of the maximum possible waiting time and the waiting fuel consumption rate F_i^{wt} . The second term accounts for emissions during service, given by the maximum handling time H_i^- multiplied by the service fuel consumption rate F_i^{ht} . The third term (E_i^-) represents the maximum emissions associated with crane operations.

The fuel consumption rates are computed as follows: $F_i^{wt} = \epsilon \cdot F^{SC} \cdot Q_i \cdot Q_i^f$, $F_i^{ht} = 0.3 \cdot F_i^{wt}$. Here, ϵ is a scaling parameter, F^{SC} denotes the specific fuel consumption of the auxiliary engines, and Q_i is the total cargo quantity of vessel i , expressed in Twenty-foot Equivalent Units (TEUs). The parameter Q_i^f represents the average container load factor, i.e., the average fraction of capacity utilised by the containers on board. For instance, $Q_i^f = 0.8$ indicates that the containers of vessel i are, on average, filled to 80% of their capacity. The fuel consumption rate during service is conservatively defined as 30% of the consumption rate during waiting. The operational cost penalty is defined as the product of the maximum handling time and the maximum berthing cost: $P_i^c = \max_j H_i^j \cdot \max_j C^j = H_i^- \cdot C^-$ where the berthing cost at berth j is given by $C^j = \Theta_b \cdot L^j + \Theta_c \cdot N^j$. This cost consists of the berth reservation fee multiplied by the berth length and the crane operation fee multiplied by the number of cranes assigned to that berth.

Finally, the load-related penalty P^l provides a conservative upper bound on workload imbalance across the terminal. It is defined as the variance of berth loads under a worst-case assignment in which all vessels are hypothetically allocated to the slowest berth j_0 . This extreme scenario induces uniformly large handling times and maximizes load imbalance across berths, yielding an upper-bound reference for load dispersion. Let H_i^- denote the handling time of vessel i when processed at berth j_0 . Under this assumption, the load penalty is computed as $P^l = \frac{|B|-1}{|B|} \left(\sum_{i \in S} H_i^- \right)^2$.

B Baseline vs. Adaptive Variants

Table 5 Baseline vs. Adaptive Variants Experiment. Red boldface indicates the best value of the metric for the benchmark set.

Algorithm	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓	T(s)	HV↑	Δ↓	IGD↓
40-10				45-10				60-20				
NSGAII	17.7	1.085	0.699	0.045	23.3	1.098	0.733	0.058	59.7	1.063	0.818	0.046
NSAGAI	13.9	1.098	0.693	0.039	18.6	1.130	0.701	0.048	50.3	1.069	0.779	0.038
NSGAIII	17.5	1.065	0.836	0.049	23.3	1.082	0.753	0.052	59.3	1.051	0.812	0.043
NSAGAI	14.1	1.084	0.832	0.039	18.8	1.117	0.818	0.040	51.0	1.052	0.942	0.043
SPEA2	18.2	1.063	0.758	0.080	24.0	1.081	0.763	0.087	60.1	1.054	0.836	0.057
SPAEA2	14.2	1.087	0.751	0.079	19.2	1.104	0.750	0.099	50.5	1.057	0.828	0.073
MOEA/D	18.2	1.076	0.743	0.043	24.4	1.041	0.753	0.063	57.0	1.064	0.650	0.038
MOEA/D	15.0	1.091	0.755	0.027	20.1	1.120	0.697	0.034	51.7	1.065	0.641	0.036
70-20				80-20				90-20				
NSGAII	82.5	1.094	0.707	0.043	108.7	1.166	0.779	0.031	133.2	1.136	0.694	0.047
NSAGAI	66.1	1.107	0.705	0.043	85.5	1.183	0.781	0.028	107.3	1.158	0.713	0.040
NSGAIII	82.1	1.072	0.855	0.040	108.8	1.160	0.950	0.031	133.2	1.132	0.713	0.035
NSAGAI	67.6	1.102	0.757	0.035	86.6	1.164	0.789	0.029	106.7	1.139	0.766	0.040
SPEA2	82.3	1.084	0.900	0.066	108.7	1.175	0.871	0.041	133.4	1.125	0.804	0.067
SPAEA2	67.1	1.099	0.789	0.057	86.3	1.178	0.828	0.039	107.5	1.147	0.788	0.046
MOEA/D	79.1	1.052	0.693	0.043	102.6	1.152	0.688	0.059	120.2	1.046	0.714	0.060
MOEA/D	67.6	1.085	0.666	0.026	88.9	1.157	0.700	0.040	103.9	1.109	0.745	0.046
100-25				120-30				135-30				
NSGAII	205.4	1.084	0.698	0.037	316.7	1.088	0.698	0.044	407.0	1.045	0.700	0.053
NSAGAI	164.2	1.100	0.736	0.036	248.9	1.105	0.749	0.040	317.0	1.057	0.694	0.050
NSGAIII	204.9	1.076	0.721	0.032	316.9	1.069	0.710	0.040	403.5	1.048	0.668	0.044
NSAGAI	166.7	1.082	0.904	0.034	251.9	1.089	0.877	0.041	322.3	1.064	0.801	0.042
SPEA2	204.1	1.080	0.748	0.059	316.9	1.088	0.842	0.064	405.5	1.046	0.791	0.067
SPAEA2	164.6	1.083	0.762	0.057	250.4	1.090	0.857	0.062	319.4	1.052	0.749	0.059
MOEA/D	188.0	1.074	0.716	0.028	282.0	1.063	0.727	0.028	364.5	1.020	0.707	0.036
MOEA/D	162.0	1.089	0.696	0.024	246.3	1.090	0.696	0.025	311.3	1.048	0.674	0.032
140-35				150-35								
NSGAII	503.6	1.064	0.745	0.054	590.5	1.058	0.742	0.046				
NSAGAI	394.9	1.083	0.687	0.045	455.9	1.069	0.693	0.038				
NSGAIII	503.1	1.057	0.739	0.055	586.1	1.044	0.829	0.043				
NSAGAI	399.5	1.062	0.761	0.059	455.4	1.052	0.795	0.037				
SPEA2	504.8	1.063	0.782	0.0457	585.2	1.045	0.800	0.063				
SPAEA2	394.4	1.080	0.731	0.056	449.3	1.062	0.875	0.052				
MOEA/D	454.6	1.022	0.723	0.049	506.1	1.028	0.715	0.052				
MOEA/D	387.2	1.044	0.694	0.038	439.7	1.036	0.749	0.031				

Having identified **RandomParent** as the most effective crossover operator on average, we evaluate the impact of the **ALNS**-inspired adaptive operator selection mechanism on classical MOEAs. We compare each baseline algorithm, configured with the **RandomParent** crossover operator and the **ReInsert** mutation operator, against its corresponding adaptive variant, which dynamically selects among all available crossover and mutation operators.

Table 5 summarizes the comparative results, with all algorithms executed using a population size of 100 for 200 generations. Overall, the adaptive variants demonstrate stronger performance across most problem instances and evaluation metrics. A notable improvement appears in computational time: in all instances, the adaptive variants are substantially faster

than their non-adaptive counterparts. This reduction can be attributed to the adaptive operator selection mechanism, which enables the use of multiple crossover and mutation operators, including operators that are faster than **RandomParent** and **ReInsert**. In terms of solution quality, the adaptive variants achieve in most cases, higher HV, lower Δ and IGD.

C Non-Dominated Solutions

In this section, we present the results of our experimental evaluation on the same benchmark instances and under the same termination criteria described in Section 4, using the **Recall** metric. Recall measures the fraction of solutions in the reference Pareto front that are contributed by a given algorithm. We consider this metric because it is particularly sensitive to the number of non-dominated solutions returned by each algorithm.

Four algorithms, namely **MOAEA/D**, **NSACSA/D**, **EMOABC**, and **SMOALNS**, maintain an external archive of non-dominated solutions. Consequently, the number of solutions they return is not fixed, unlike the remaining algorithms, which always return a population of predetermined size. We therefore investigate how this characteristic affects the Recall metric.

As shown in Table 6, **MOAEA/D** consistently achieves the highest Recall values. In the same generations experiment, **SMOALNS** ranks second in 10 out of the 11 cases, with **NSACSA/D** outperforming it only for case 90–20. **NSALNS** consistently ranks behind these three algorithms, except in the 45–10, 120–30, and 150–35 cases, where it outperforms **NSACSA/D**, while it consistently outperforms all remaining methods. Among the evolutionary algorithms with a fixed population size, **NSGAI**, **NSGAI**, and **SPEA2** achieve the lowest Recall values. Specifically, **NSGAI** outperforms **NSGAI** in 10 of the 11 cases, whereas **NSGAI** is superior only once and outperforms **SPEA2** in all but the 140–35 and 150–35 cases. These results are expected, as the archive-based design of **MOAEA/D**, **SMOALNS**, and **NSACSA/D** enables them to contribute substantially more non-dominated solutions to the reference Pareto front than algorithms that return a fixed-size population. In contrast, although **EMOABC** also maintains an external archive, it consistently achieves the lowest Recall values because the quality of its generated solutions is considerably lower than that of the competing methods, resulting in only a limited contribution to the reference Pareto front.

The results obtained under the time-limit criterion, reported in Table 7, exhibit a different ranking among algorithms. Although **MOAEA/D** again achieves the highest Recall values, **NSALNS** now outperforms **NSACSA/D** in all but four cases and performs comparably to **SMOALNS**, outperforming it in five cases while ranking behind it in six. Furthermore, **NSACSA/D** achieves the second-highest Recall value only in three cases. Among the fixed-population evolutionary algorithms, **NSGAI** outperforms **NSGAI** in all but the 45–10 instance and even surpasses **NSACSA/D** for cases 40–10, 80–20 and 90–20, as well as **SMOALNS** for cases 90–20, 135–30, and 150–35. **SPEA2** consistently achieves lower Recall values than both **NSGAI** and **NSGAI**, while **EMOABC** again obtains the lowest Recall values overall. The different ranking observed under the time-limit criterion can be explained by the computational cost of the algorithms. As discussed in Section 4, the time limit is set equal to the average execution time of **NSALNS** for each instance, as it is the best-performing algorithm in HV and Δ . **NSACSA/D** and **SMOALNS** require substantially more computation per generation than **NSALNS**. Consequently, under a fixed computational budget they complete fewer generations, reducing the number of non-dominated solutions they contribute to the reference Pareto front and, therefore, their Recall values.

An interesting observation is that **NSALNS**, despite achieving the best overall performance in terms of HV and Δ , does not return the largest number of non-dominated solutions. Nevertheless, it produces the second-largest solution sets in several instances of the Time

■ **Table 6** Same Generations Experiment Recall.

Algorithm	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R
	40-10		45-10		60-20		70-20		80-20		90-20	
NSAGAI	13.4	0.051	18.0	0.047	50.0	0.048	65.4	0.046	84.0	0.055	105.8	0.046
NSAGAI	13.5	0.052	18.4	0.039	50.9	0.058	66.3	0.058	85.6	0.059	104.8	0.074
SPAEA2	13.8	0.034	18.6	0.033	50.1	0.038	66.0	0.036	85.5	0.031	106.5	0.032
MOAEA/D	14.1	0.491	19.3	0.569	50.5	0.447	66.0	0.471	85.3	0.447	101.4	0.545
EMOABC	7.5	0.001	10.7	0.001	43.6	0.001	69.9	0.003	95.0	0.002	136.7	0.001
NSACSA/D	27.1	0.083	35.5	0.079	101.9	0.118	135.5	0.122	178.9	0.121	214.2	0.112
SMOALNS	12.2	0.211	16.9	0.152	50.2	0.188	71.4	0.182	104.9	0.190	141.7	0.111
NSALNS	11.3	0.077	15.8	0.081	45.9	0.103	66.6	0.082	97.7	0.095	133.4	0.080
Algorithm	100-25		120-30		135-30		140-35		150-35			
NSAGAI	162.8	0.031	249.4	0.038	316.3	0.043	391.0	0.042	464.0	0.047		
NSAGAI	163.8	0.051	251.3	0.073	321.1	0.078	397.7	0.076	467.0	0.084		
SPAEA2	161.8	0.024	248.6	0.027	314.7	0.027	390.5	0.043	464.8	0.054		
MOAEA/D	158.9	0.528	247.2	0.539	305.3	0.534	388.8	0.414	455.5	0.452		
EMOABC	186.8	0.001	299.6	0.001	394.9	0.002	473.7	0.004	547.5	0.003		
NSACSA/D	333.9	0.143	523.1	0.082	656.9	0.098	848.1	0.096	950.6	0.092		
SMOALNS	223.0	0.150	385.5	0.145	562.2	0.131	680.1	0.232	813.2	0.173		
NSALNS	208.7	0.071	356.8	0.095	514.7	0.085	636.0	0.093	740.4	0.093		

Limit Experiment and, among all algorithms with a fixed output size, consistently attains the highest Recall values across both experimental settings. These results further demonstrate that the quality and distribution of the generated solutions are more influential than their quantity with respect to the HV and Δ indicators.

By contrast, IGD is inherently more sensitive to the density of solutions approximating the reference Pareto front. Consequently, algorithms that contribute a larger number of high-quality non-dominated solutions tend to achieve better IGD values. This explains why SMOALNS outperforms NSALNS in terms of IGD in the experimental evaluation presented in Section 4, despite the superior HV and Δ performance of NSALNS.

■ **Table 7** Time Limit Experiment Recall.

Algorithm	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R	T(s)	R
	40-10		45-10		60-20		70-20		80-20		90-20	
NSAGAI		0.052		0.049		0.056		0.051		0.049		0.057
NSAGAI		0.063		0.044		0.070		0.060		0.059		0.088
SPAEA2		0.042		0.030		0.043		0.034		0.030		0.053
MOAEA/D	12	0.560	16	0.650	45	0.519	67	0.564	98	0.593	134	0.599
EMOABC		0.001		0.001		0.004		0.003		0.001		0.001
NSACSA/D		0.058		0.053		0.071		0.081		0.057		0.039
SMOALNS		0.117		0.076		0.113		0.106		0.112		0.065
NSALNS		0.105		0.097		0.125		0.101		0.099		0.099
Algorithm	100-25		120-30		135-30		140-35		150-35			
NSAGAI		0.022		0.041		0.039		0.032		0.041		
NSAGAI		0.041		0.060		0.054		0.058		0.078		
SPAEA2		0.023		0.028		0.033		0.034		0.042		
MOAEA/D	209	0.633	357	0.643	515	0.619	636	0.594	741	0.566		
EMOABC		0.001		0.002		0.001		0.003		0.003		
NSACSA/D		0.128		0.062		0.132		0.096		0.113		
SMOALNS		0.083		0.088		0.048		0.110		0.073		
NSALNS		0.069		0.076		0.074		0.072		0.084		