A Reinforcement Learning Approach for the Dynamic Berth Allocation Problem

Carlos March Moya[1(✉)], Frederik Schulte[1], Kevin Tierney[2], and Neil Yorke-Smith[1]

[1] Delft University of Technology, Delft, The Netherlands
{c.marchmoya,f.schulte,n.yorke-smith}@tudelft.nl
[2] Department of Business Decisions and Analytics, University of Vienna, Vienna, Austria
kevin.tierney@univie.ac.at

Abstract. Efficiently assigning vessels to berths in a busy port is a hard scheduling problem: decisions must be made in real time, under uncertainty about future arrivals, and with the goal of minimizing how long ships spend in the port. We present a learning-based approach in which an agent is trained, purely through trial and error, to perform these assignments one at a time as the vessels arrive. The agent represents the current port state as a graph, capturing relationships between waiting vessels, available berths, and ships still on their way, and learns which assignments are promising via graph-based attention. Our formulation incorporates vessel priorities, berth-dependent handling times, and a skip action that allows the agent to postpone assignments when beneficial. Across a range of port congestion levels, the trained agent outperforms classical heuristic rules, achieving on average a 20% reduction in priority-weighted turnaround cost relative to the best heuristic baseline.

Keywords: Berth Allocation Problem · Reinforcement Learning · Graph Neural Networks · Operational Research

1 Introduction

Berth Allocation represents one of the bottlenecks in maritime logistics, significantly increasing the time that vessels spend in virtual queues and, therefore, contributing substantially to emissions associated with port operations [10]. Indeed, emissions associated with port calls and anchorage waiting can reach levels comparable to the annual emissions of entire countries, as reported in the case of the Port of Rotterdam. This highlights the critical role of efficient berth allocation not only in improving port performance but also in advancing decarbonization efforts within maritime transport.

Efficient and intelligent port logistics are therefore essential to ensure the smooth operation of global supply chains while reducing both operational costs

J. de Armas and E. Lalla Ruiz (Eds.): LOGMS 2026, LNCS 16679, pp. 15–26, 2027.
https://doi.org/10.1007/978-3-032-30795-8_2

and environmental impact. Within this context, one of the core operational optimization problems is the Berth Allocation Problem (BAP), which consists of deciding to which berth each arriving vessel should be assigned and at what time, taking into account berth-dependent handling times and operational constraints. When vessels arrive dynamically over time rather than being in the port at the beginning of the planning horizon, the problem is referred to as the Dynamic Berth Allocation Problem (DBAP).

The BAP is an NP-hard problem [4], and its dynamic nature makes real-time decision-making particularly challenging. In practice, ports often rely on simple dispatching rules due to their speed and ease of implementation. However, these approaches typically fail to capture the global structure of the problem and may lead to suboptimal performance.

In recent years, learning-based optimization methods have emerged as a promising alternative to custom heuristics and metaheuristics. In particular, reinforcement learning (RL) provides a natural framework for modelling sequential decision-making under uncertainty, enabling agents to learn policies directly from interaction with the environment. Nevertheless, its application to maritime logistics problems remains relatively limited. Particularly, for dynamic scheduling problems, (deep) reinforcement learning approaches seem to offer advantages that are less explored structurally similar problems as the dynamic BAP.

Motivated by these challenges, this paper proposes a size-agnostic, graph-based reinforcement learning framework for the discrete BAP with vessel priorities and berth-dependent handling times. The main contributions of this work are as follows:

1. We introduce an MDP formulation of the discrete BAP that incorporates vessel priorities, berth-dependent handling times, and a skip action allowing the agent to defer assignments when beneficial.
2. We propose a Graph Neural Network (GNN)-based architecture capable of generalizing across instances with varying numbers of vessels and berths, integrating both local and global contextual information.
3. We conduct an empirical evaluation on 2700 randomly generated instances, demonstrating that the proposed method consistently outperforms classical dispatching heuristics.

2 Related Work

The BAP is classified along two axes: spatial layout (discrete, continuous, or hybrid) and temporal setting (static, when all vessels are known upfront, or dynamic, when they arrive over time). This work addresses the discrete dynamic variant (DDBAP). Early exact approaches for the DDBAP include a Lagrangian relaxation method [5] and subsequent Mixed-Integer Programming (MIP) extensions [3]; while optimal for small instances, these methods do not scale. Metaheuristics such as hybrid genetic algorithms [1] and bee colony optimization [9] improve scalability but still require per-instance computation and do not generalize across problem configurations.

In practice, ports rely on simple dispatching heuristics for their speed, but these ignore global scheduling structure and yield large optimality gaps under congestion. This has motivated the use of reinforcement learning to learn decision policies directly from experience. RL has shown strong results in combinatorial optimization (routing, scheduling), yet its application to maritime logistics remains limited. In [8], the agent selects among predefined heuristics at each step to resolve conflicts, using a fixed-size state representation tied to a specific number of berths. More recently, [6] propose a hierarchical multi-agent MDP for joint vessel selection, berth allocation, and crane assignment, while [7] train a size-agnostic model via evolutionary strategies for the continuous BAP.

Only [7] and [6] propose size-agnostic approaches, but neither fully addresses our setting: [7] lacks a skip action and explicit priority modelling, and [6] encodes only a simplified state without vessel priorities. In contrast, this work proposes a unified MDP with a graph-based size-agnostic state representation, explicit priority modelling, and a skip action, enabling robust generalization across varying port configurations.

3 Problem Formulation

We address the Dynamic Discrete Berth Allocation Problem (DDBAP) with vessel priorities and berth-dependent handling times. A port has a set of discrete berths $\mathcal{B} = \{1, \ldots, B\}$ and receives a set of vessels $\mathcal{V} = \{1, \ldots, V\}$ dynamically over a rolling planning horizon. Each vessel v is characterized by its arrival time $a_v \geq 0$, a berth-dependent handling time $h_{v,b} > 0$ for each berth $b \in \mathcal{B}$, and an integer priority weight $p_v \in \mathbb{Z}_{\geq 1}$. Four families of decision variables are introduced: the continuous start time $t_v \geq 0$; the binary berth-assignment $x_{v,b} \in \{0, 1\}$; the binary sequencing variable $y_{v,v'} \in \{0, 1\}$, equal to 1 iff v is served *immediately before* v' at the same berth; and the first/last-in-berth indicators $f_v, l_v \in \{0, 1\}$.

The resulting MIP model is adapted to consider the service time from [8]:

$$\min \quad \sum_{v \in \mathcal{V}} p_v \left(t_v - a_v + \sum_{b \in \mathcal{B}} h_{v,b}\, x_{v,b} \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{b \in \mathcal{B}} x_{v,b} = 1 \qquad \forall v \in \mathcal{V} \tag{2}$$

$$f_v + \sum_{\substack{v' \in \mathcal{V} \\ v' \neq v}} y_{v',v} = 1 \qquad \forall v \in \mathcal{V} \tag{3}$$

$$l_v + \sum_{\substack{v' \in \mathcal{V} \\ v' \neq v}} y_{v,v'} = 1 \qquad \forall v \in \mathcal{V} \tag{4}$$

$$f_v + f_{v'} \leq 3 - x_{v,b} - x_{v',b} \qquad \forall b \in \mathcal{B},\ \forall v, v' \in \mathcal{V},\ v \neq v' \tag{5}$$

$$l_v + l_{v'} \leq 3 - x_{v,b} - x_{v',b} \qquad \forall b \in \mathcal{B},\ \forall v, v' \in \mathcal{V},\ v \neq v' \tag{6}$$

$$y_{v,v'} - 1 \leq x_{v,b} - x_{v',b} \leq 1 - y_{v,v'} \qquad \forall b \in \mathcal{B},\ \forall v, v' \in \mathcal{V},\ v \neq v' \tag{7}$$

$$t_v + \sum_{b \in \mathcal{B}} h_{v,b}\, x_{v,b} - M(1 - y_{v,v'}) \leq t_{v'} \qquad \forall v, v' \in \mathcal{V},\ v \neq v' \tag{8}$$

$$t_v \geq a_v \qquad \forall v \in \mathcal{V} \tag{9}$$

The objective function (1) minimizes total priority-weighted service time. Constraints (2) enforces a unique berth per vessel. Constraints (3)–(6) link sequencing variables to first/last-in-berth indicators, ensuring each berth has exactly one head and one tail. Constraints (7) couples ordering and assignment ($y_{v,v'} = 1$ only if v, v' share a berth), Constraints (8) enforces non-overlap via big-M, and Constraints (9) ensure service cannot start before arrival.

4 Markov Decision Process

In order to solve the DDBAP using reinforcement learning we formulate it as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where decisions are made sequentially as vessels arrive and berths become available.

4.1 State Space

At each decision step t, the state $s_t \in \mathcal{S}$ is represented by a single heterogeneous bipartite graph $G = (\mathcal{V}, \mathcal{B}, E)$ containing all vessels (both waiting and not-yet-arrived) and all berths as nodes, with edges whose features encode assignment costs and vessel status.

Vessel Nodes. Each vessel $v \in \mathcal{V}$ is described by a four-dimensional feature vector:

$$\mathbf{f}^{(v)} = \left[\, \tilde{w}_v,\ \tilde{a}_v,\ \tilde{p}_v,\ \tilde{q}_v \,\right], \tag{10}$$

where $\tilde{w}_v = \max(0,\ t_{\text{cur}} - a_v)/T$ is the normalized waiting time already accumulated (0 for vessels not yet arrived), $\tilde{a}_v = \max(0,\ a_v - t_{\text{cur}})/T$ is the normalized time until arrival (0 for arrived vessels), $\tilde{p}_v = p_v/p_{\max}$ is the normalized priority, and $\tilde{q}_v = \max_b(p_v/h_{v,b})/q_{\max}$ is the vessel's best achievable score across all berths, with T as the planning horizon, $p_{\max}$ as the maximum priority and $q_{\max}$ as the maximum weighted processing time and priority in the instance. Exactly one of $\tilde{w}_v$ and $\tilde{a}_v$ is non-zero at any time, encoding whether the vessel is already waiting or is still in transit

Berth Nodes. Each berth $b \in \mathcal{B}$ carries a three-dimensional feature vector:

$$\mathbf{f}^{(b)} = \left[\, \tilde{r}_b,\ \tilde{q}_b,\ o_b \,\right], \tag{11}$$

where $\tilde{r}_b = \max(0,\ f_b - t_{\text{cur}})/T$ is the normalized remaining time until the berth is free, $\tilde{q}_b = \max_v(p_v/h_{v,b})/q_{\max}$ is the best score achievable at this berth across all vessels, and $o_b \in \{0, 1\}$ is an occupancy indicator.

Edge Features. Every pair of (vessel, berth) is connected by an edge $(v, b) \in E$ with a six-dimensional feature vector:

$$\mathbf{e}_{vb} = \left[\, \tilde{s}_{vb},\ \tilde{c}_{vb},\ \tilde{h}_{vb},\ \mathbb{1}_{\text{valid}},\ \mathbb{1}_{\text{skip}},\ \mathbb{1}_{\text{future}} \,\right], \tag{12}$$

where $\tilde{s}_{vb} = (p_v/h_{v,b})/q_{\max}$ is the normalized ranking score for this (vessel, berth) pair, $\tilde{c}_{vb} = (\max(0,\ t_{\text{cur}} - a_v) \cdot p_v)/\mathcal{L}$ is the weighted waiting cost already accumulated by vessel v, normalized by a reference solution value $\mathcal{L}$, and the last three dimensions form a mutually exclusive one-hot status encoding: *valid* means the vessel has arrived and the edge is not forbidden (immediately actionable), *skip* means the vessel has arrived but this pair was skipped and is temporarily blocked, and *future* means the vessel has not yet arrived. Figure 1 illustrates the graph representation of a state in which three vessels have arrived, two more are coming and one of three berths is occupied. Due to a prior `Skip` action, vessel 2 cannot be assigned to berth 1.

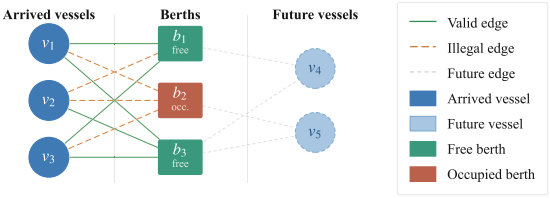

Fig. 1. Unified bipartite state graph example.

4.2 Action Space

At each state s_t, the agent selects an action from the set $\mathcal{A}(s_t) = \mathcal{A}^{\text{assign}}(s_t) \cup \{\texttt{Skip}\}$. Each action has a different impact on the environment.

Assignment Actions. A feasible assignment action is a pair (v, b) where v is an arrived vessel (i.e. $a_v \le t_{\text{cur}}$) and $b \in \mathcal{B}$ is a free berth such that the corresponding edge (v, b) carries status $\mathbb{1}_{\text{valid}} = 1$ (not forbidden by a prior skip).

Skip Action. The agent may also choose to `Skip`, advancing the clock to the next relevant event, either a berth becoming free or a new vessel arriving, without making an assignment. This is necessary because it can be beneficial to wait for a future vessel before committing to an assignment.

Forbidden-Edge Masking. To make the `Skip` action learnable, whenever the `Skip` is taken, every edge (v, b) that is available at that step (vessel waiting, berth free) is marked *forbidden* for all subsequent states until either that vessel is assigned or berth b is re-occupied. When a vessel is assigned, all forbidden edges involving that vessel or berth are cleared. This encodes the constraint that re-considering a declined pair at a later time can only worsen the objective. If no future vessels remain and all berths are free, the forbidden set is cleared and `Skip` is masked. Additionally, if an available assignment dominates `Skip` (the vessel could be fully processed before the next event), `Skip` is also masked.

Transition. In an assignment of vessel v to berth b, the vessel node is removed from the graph and the time advances to $t_{\text{next}} = \max\big(\min_{v'}(a_{v'}), \min_{b'}(f_{b'}), t_{cur}\big)$, where $\min_{v'}(a_{v'})$ is the earliest arrival among unassigned vessels not yet in port and $\min_{b'}(f_{b'})$ is the earliest time any occupied berth becomes free. In other words, the next decision point is as soon as *either* a new vessel is available *or* a berth opens up, whichever comes first, but never before the current time. On a `skip`, the vessel nodes remain unchanged and the clock advances only to the *next event*: whichever of (next vessel arrival, next berth becoming free) happens first strictly after t_{cur}, since by construction there is already at least one waiting vessel and one free berth at the time of the skip.

4.3 Reward

The reward signal is an incremental penalty based on changes in the objective function, providing a dense learning signal throughout each episode.

Assignment Reward. When the agent assigns vessel v to berth b at time t:

$$r_t^{\text{assign}} = -\frac{p_v h_{v,b} + \Delta W_t}{\mathcal{L}}, \tag{13}$$

where $p_v h_{v,b}$ is the priority-weighted handling cost, $\Delta W_t = \sum_{v' \neq v} p_{v'} \Delta w_{v',t}$ is the additional waiting-time cost incurred by queued vessels when the clock advances. Waiting time is tracked incrementally via a counter $\hat{w}_v$; only the new increment $\Delta w_{v,t} = (t_{\text{cur}} - a_v) - \hat{w}_v$ is taken into account per step to avoid double counting.

Skip Reward. When the agent takes `Skip`, advancing the clock by Δt:

$$r_t^{\text{skip}} = -\frac{\Delta W_t}{\mathcal{L}} \tag{14}$$

In both cases, $\mathcal{L}$ is a reference solution value that stabilizes training by scaling rewards across instances of different sizes.

5 Proposed Method

In this section, we present the proposed reinforcement learning approach for solving the berth allocation problem. The architecture is based on a Graph Neural Network, which encodes the problem as a bipartite graph and guides the decision-making process.

5.1 GNN Architecture

The agent consists of an encoder that processes the unified bipartite graph G (containing all vessels and all berths) and a decoder that assigns scores to candidate assignment actions via a two-branch edge scorer.

GNN Encoder. The graph is processed by a encoder consisting of $L = 3$ heterogeneous graph convolution layers built with Graph ATtention networks (GAT) [2] operating on the bipartite graph. At each layer ℓ, messages are passed in both directions (vessel $\rightarrow$ berth and berth $\rightarrow$ vessel):

$$\mathbf{h}_v^{(\ell)} = \mathrm{ELU}\Big(\mathrm{LN}\Big(\mathrm{GAT}_{(\ell)}^{v\to b}\big(\mathbf{h}_v^{(\ell-1)}, \mathbf{h}_b^{(\ell-1)}, \mathbf{e}_{vb}\big)\Big)\Big), \tag{15}$$

$$\mathbf{h}_b^{(\ell)} = \mathrm{ELU}\Big(\mathrm{LN}\Big(\mathrm{GAT}_{(\ell)}^{b\to v}\big(\mathbf{h}_b^{(\ell-1)}, \mathbf{h}_v^{(\ell-1)}, \mathbf{e}_{vb}\big)\Big)\Big), \tag{16}$$

where $\mathbf{e}_{vb} \in \mathbb{R}^5$ is the edge feature vector defined in Eq. (12), LN denotes Layer Normalization, and ELU is the Exponential Linear Unit activation. Residual connections are added from layer 2 onwards (once the feature dimension stabilises at d_h). All attention layers use $H = 4$ heads with mean aggregation, yielding output dimension d_h. The first layer projects raw features (vessel: $\mathbb{R}^4$, berth: $\mathbb{R}^3$) into $\mathbb{R}^{d_h}$; subsequent layers operate entirely in $\mathbb{R}^{d_h}$. The encoder produces:

- Per-node embeddings $\mathbf{H} \in \mathbb{R}^{(|\mathcal{V}|+|\mathcal{B}|)\times d_h}$ (vessels concatenated with berths).
- A global graph embedding $\mathbf{g} \in \mathbb{R}^{d_h}$ obtained via mean pooling over all nodes.

Edge Scoring (Actor). For each candidate vessel–berth pair (v, b) with edge status $\mathbb{1}_{\text{valid}} = 1$, a score is computed by summing two complementary branches:

$$\mathrm{score}(v,b) = \underbrace{\mathrm{MLP}_{\text{local}}\big(\mathbf{e}_{vb}^{(1:3)}\big)}_{\text{local ranking}} + \underbrace{\mathrm{MLP}_{\text{ctx}}\big([\,\mathbf{h}_v^{(L)} \,\|\, \mathbf{h}_b^{(L)}\,]\big)}_{\text{context}}, \tag{17}$$

where $\mathbf{e}_{vb}^{(1:3)} = [\tilde{s}_{vb}, \tilde{c}_{vb}, \tilde{h}_{vb}]$ are the first three (numeric) edge features. The *local branch* $\mathrm{MLP}_{\text{local}}$ has layers $[3 \to d_h \to 1]$ and captures the immediate assignment quality signal (WTSP score and accumulated waiting cost). The *context branch* $\mathrm{MLP}_{\text{ctx}}$ has layers $[2d_h \to d_h \to 1]$ and captures interactions such as berth congestion and future vessel arrivals encoded in the GNN embeddings. An additional `Skip` score $\mathrm{score}_{\text{skip}} \in \mathbb{R}$ is produced by a separate MLP with layers $[d_h \to d_h \to d_h/2 \to 1]$ taking the global graph embedding $\mathbf{g}$ as input.

Action-Masking and Distribution. Before computing the softmax distribution, all logits corresponding to invalid actions (occupied berths, skipped edges, future-vessel edges, or `Skip` when no future vessels remain and all berths are free) are set to $-\infty$. During training, an action is sampled from the policy; at evaluation, the argmax is taken (deterministic greedy policy). This design is *size-agnostic*: the GNN encoder operates on graphs of arbitrary size, so the same

model can be applied to instances with different numbers of vessels and berths without re-training. For the trained model the hidden dimension d_h is set to 64.

Value Network (Critic). A separate value network estimates $V(s_t) = \mathrm{MLP}_{\mathrm{value}}(\mathbf{g})$ with layers $[d_h \to d_h \to d_h/2 \to 1]$ and ELU activations.

5.2 Training Procedure

The agent is trained with Proximal Policy Optimization. Training proceeds in batches: at the start of each batch, a fixed random seed is set and N_{inst} BAP instances are generated. Episodes are distributed across instances in round-robin fashion according to E_{inst}, interleaving learning updates across different problem realizations. For each episode the agent collects a full trajectory and computes discounted returns and GAE advantages ($\gamma = 0.99$, $\lambda = 0.95$), normalized to zero mean and unit variance. The actor and critic are then updated for $K = 4$ epochs using the clipped surrogate objective ($\epsilon = 0.2$), with a value loss coefficient $c_1 = 0.5$ and entropy bonus $c_2 = 0.01$. Gradients are clipped to a max norm of 0.5 and Adam is used with learning rate $\alpha = 10^{-4}$. The final model is selected as the checkpoint with the best performance on a held-out validation set.

6 Experimental Study

In this section, we first describe how the instances used for training, validation, and testing were generated. We then present the performance of the trained agent on both the validation and test sets, comparing it against an exact solver (when feasible within the imposed time limits) and four dispatching heuristics.

Instance Generation. All instance sets (training, validation, test) share the same generation procedure. Vessel arrivals follow the methodology of [8]: inter-arrival times are sampled from an exponential distribution with rate $\lambda = 1/I$, where $I = C \cdot \frac{\sum_{v,b} h_{v,b}}{BV^2}$ and $C > 0$ controls congestion (smaller C = higher congestion). Handling times are drawn from $\mathcal{U}[10, 200]$ and priorities from $\{1, \ldots, 10\}$. Training batches alternate between small (10 vessels, 3 berths) and medium (30 vessels, 5 berths) instances across $C \in \{0.5, 1.0, 2.0\}$, generating 12 instances per batch with $E_{\mathrm{inst}} = 5$ episodes each. The validation set contains six fixed instances (same sizes/congestions as training). Test instances cover all 27 combinations of $V \in \{80, 100, 120\}$, $B \in \{5, 10, 15\}$, and $C \in \{0.5, 1.0, 2.0\}$, with 100 independent replicates per combination (2 700 instances total).

Baselines. For a fair comparison, we evaluate the agent against four dispatching heuristics and additionally against the MIP solver Gurobi (1800 s timeout). **FCFS** assigns the earliest-arrived vessel to the available berth with the shortest processing time. **SPT** always selects the vessel-berth pair with the smallest handling time. **Priority** dispatches vessels in decreasing priority order, assigning each to the available berth with the shortest processing time. **WTSP** ranks all

arrived-vessel/free-berth pairs by $p_v/h_{v,b}$ and selects the highest-scoring pair; this heuristic is used as the reference solution $\mathcal{L}$ for reward scaling. Regarding the MIP solver Gurobi, we allow 1800 s to evaluate how closely the method approaches optimal solutions. However, ports are extremely dynamic environments, requiring frequent re-optimization, which makes exact solvers like Gurobi impractical for real-time deployment. In contrast, both the heuristics and our agent can handle common sources of uncertainty in port operations, such as vessel delays, variable handling times, and other stochastic events, while requiring low computational time compared to MIP solvers, making them more suitable for operational use.

Table 1. Objective function and gap with respect to the best solution found by Gurobi of the heuristics and the agent on the validation set. † Gurobi time-limit reached.

Instance	Gurobi	FCFS	SPT	Priority	WTSP	Agent
1_10v3b_C05	306.50	+136%	+23%	+27%	+9%	**+5.2%**
2_10v3b_C1	512.50	+122%	+85%	+157%	+77%	**+3.5%**
3_10v3b_C2	449.40	+71%	+43%	+60%	+10%	**+0.0%**
4_30v5b_C05	329.47†	+244%	+59%	+189%	+69%	**+2.3%**
5_30v5b_C1	400.03†	+289%	+85%	+254%	+73%	**+3.6%**
6_30v5b_C2	414.07†	+164%	+32%	+135%	+33%	**+13%**

Results on the Validation Set. Table 1 reports the objective value and gap with respect to Gurobi for each validation instance. The agent outperforms all four heuristics on every instance; on instance 3 ($C = 2$) it recovers the proven optimal. On instances 4–6 (30 vessels) Gurobi hits its 1800 s time limit yet the agent remains competitive with or below that bound. WTSP is the strongest heuristic baseline, but the learned policy improves further by leveraging global context from the GNN embeddings.

Results on the Test Set. Table 2 reports the mean gap (%) with respect to the WTSP objective across 100 replicated instances for each of the 27 configurations. WTSP is the best-performing heuristic and is therefore used as the baseline for computing the gaps, instead of Gurobi, which does not produce feasible solutions for all instances within the timeout even though in some configurations the performance of Gurobi is better than the performance of the Agent. In addition, the table indicates, in parentheses, the percentage of instances for which Gurobi finds a feasible solution within the time limit. It is worth noting that Gurobi, with the given time, was only able to find the optimal solution for one instance. The proposed agent achieves the lowest or nearly the lowest gap in almost every configuration, consistently improving over standard heuristics. The relative improvement of the agent is more pronounced in less congested

Table 2. Mean gap (%) with respect to WTSP across 100 replicate instances per configuration. Best result per row in bold.

C	Vessels	Berths	FCFS	SPT	Priority	Agent	Gurobi	
0.5	80	5	264	13.2	162	**–9.4**	–8.2	(97%)
		10	319	6.9	214	–20.3	**–23.5**	(99%)
		15	280	1.5	193	–26.9	**–32.9**	(100%)
	100	5	283	13.5	176	**–7.8**	1.7	(85%)
		10	359	10.1	239	**–17.8**	–3.6	(82%)
		15	379	6.1	254	**–19.7**	–16.2	(95%)
	120	5	294	14.3	185	**–5.8**	27.2	(45%)
		10	401	11.4	265	**–13.5**	25.0	(52%)
		15	416	8.9	285	**–18.3**	3.1	(76%)
1.0	80	5	284	12.2	179	–9.2	**–11.4**	(100%)
		10	314	2.5	204	–24.6	**–32.5**	(100%)
		15	226	–4.0	163	–36.4	**–47.5**	(100%)
	100	5	303	13.9	191	**–7.7**	1.5	(94%)
		10	385	8.0	254	**–20.3**	–15.8	(87%)
		15	332	–1.1	226	–31.7	**–36.0**	(100%)
	120	5	312	15.3	195	**–6.5**	49.6	(43%)
		10	421	9.8	273	**–16.8**	22.4	(68%)
		15	417	2.3	282	**–27.3**	–2.9	(86%)
2.0	80	5	313	12.2	196	–10.3	**–14.0**	(99%)
		10	298	–3.3	192	–29.9	**–45.6**	(100%)
		15	99.3	–6.1	70.6	–46.1	**–56.7**	(100%)
	100	5	333	14.4	202	**–9.3**	2.4	(98%)
		10	393	0.8	261	–26.1	**–31.8**	(100%)
		15	232	–7.8	160	–41.6	**–52.6**	(99%)
	120	5	350	15.1	220	**–6.8**	81.8	(34%)
		10	468	4.1	307	**–21.3**	15.8	(70%)
		15	349	–6.3	231	–35.9	**–36.1**	(97%)
Overall mean			327	6.2	214	**–20.3**	–15.3	(85%)

settings, reaching up to 50% reduction in gap for instances with 80 vessels and 15 berths at low congestion ($C = 2.0$). In more congested scenarios, such as 120 vessels and 5 berths at $C = 0.5$, the agent still reduces the cost by roughly 6% relative to WTSP. Overall, across all instances, the agent yields a mean gap of approximately 20%, confirming its effectiveness in reducing the priority-weighted turnaround cost compared to standard heuristics.

7 Conclusion

We presented a size-agnostic graph-based reinforcement learning approach for the DDBAP. Representing the port state as a heterogeneous bipartite graph and encoding it with a multi-layer GAT, the agent captures both local assignment signals and global context without being limited by the number of vessels or berths. The MDP includes vessel priorities, berth-dependent handling times, and a skip action to defer assignments, enabling high-quality schedules. On the validation set, the agent achieves solutions close to those obtained by Gurobi when given 1800 s per instance. Experimental results on the test set, comprising 2,700 synthetic instances across 27 configurations, show that the agent consistently outperforms classical heuristics, with the largest gains observed under low congestion ($C = 2$), where greedy methods degrade the most. Across all instances, the agent achieves a mean gap of approximately 20% relative to the best heuristic. Nevertheless, the results are very promising only in synthetic data, therefore, extensive evaluation should be performed in the future, including real world data in order to evaluate the practical contribution of the proposed method. In addition, a large validation set with distributions different from the one used in training should also be considered in future works to make the model selection more robust. Finally, future work will also include extending to the continuous BAP, handling arrival-time uncertainty, and incorporating features such as shore power.

Acknowledgments. This work is part of the PortCall. Zero project within the NWO Perspective programme, project number P23.015, financed by the Netherlands Organisation for Scientific Research (NWO).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bacalhau, E.T., Casacio, L., de Azevedo, A.T.: New hybrid genetic algorithms to solve dynamic berth allocation problem. Expert Syst. Appl. **167**, 114198 (2021)
2. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks?. arXiv preprint arXiv:2105.14491 (2021)
3. Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J., Lusby, R.: Models for the discrete berth allocation problem: a computational comparison. Transp. Res. Part E: Logist. Transp. Rev. **47**(4), 461–473 (2011)
4. Hansen, P., Oguz, C.: A note on formulations of static and dynamic berth allocation problems. Les Cahiers du GERAD ISSN **711**, 2440 (2003)
5. Imai, A., Nishimura, E., Papadimitriou, S.: The dynamic berth allocation problem for a container port. Transp. Res. Part B: Methodol. **35**(4), 401–417 (2001)
6. Jo, S., Moon, I.: A hierarchical reinforcement learning approach for real-time berth allocation and quay crane scheduling. Int. J. Prod. Res. **63**(24), 10027–10052 (2025)
7. Li, S., Xu, P., Ding, Z., Kong, Z., Luo, W.: Evolutionary reinforcement learning with density-based behavioral diversity enhancement for berth allocation and crane assignment. Swarm Evol. Comput. **100**, 102241 (2026)

8. Lv, Y., Zou, M., Li, J., Liu, J.: Dynamic berth allocation under uncertainties based on deep reinforcement learning towards resilient ports. Ocean Coastal Manag. **252**, 107113 (2024)
9. Prencipe, L.P., Marinelli, M.: A novel mathematical formulation for solving the dynamic and discrete berth allocation problem by using the bee colony optimisation algorithm: A novel mathematical formulation for solving the dynamic and discrete berth allocation problem by using the bee colony optimisation algorithm. Appl. Intell. **51**(7), 4127–4142 (2021)
10. United Nations Conference on Trade and Development: Review of maritime transport 2024 (2024). https://unctad.org/publication/review-maritime-transport-2024