Graph Neural Networks and Deep Reinforcement Learning for Warehouse Order Picking and Representation Learning

Nejc Čelik * and Andrej Škraba

Posted Date: 12 February 2026

doi: 10.20944/preprints202602.0896.v1

Keywords: order picking; deep reinforcement learning; graph neural networks;graph-based representation learning; warehouse routing optimization

 doi:10.20944/preprints202602.0896.v1

Article

Graph Neural Networks and Deep Reinforcement Learning for Warehouse Order Picking and Representation Learning

Nejc Čelik * and Andrej Škraba

Cybernetics & Decision Support Systems Laboratory, Faculty of Organizational Sciences, University of Maribor, Kidričeva cesta 55a, 4000 Kranj, Slovenia

* Correspondence: nejc.celik1@um.si

Abstract

Order picking is one of the most labor-intensive warehouse operations, and improving routing efficiency remains an important challenge. While deep reinforcement learning (DRL) has shown promise in complex optimization problems, its application to warehouse order picking is still limited, and graph-based representation learning using graph neural networks (GNNs) in this context remain largely unexplored. This paper proposes a GNN-based DRL method that models warehouse layouts as graphs to optimize order-picking paths while simultaneously learning graph-based structural embeddings of storage locations. The approach is evaluated in simulated warehouse environments of different scales and benchmarked against classical heuristics, including the Lin–Kernighan algorithm. The results show that the proposed GNN–DRL approach consistently achieves shorter travel distances than traditional methods, particularly for larger orders, and remains effective across different warehouse layouts when fine-tuned. Moreover, the learned node embeddings capture meaningful structural properties of warehouse layouts and adapt to different operational contexts, highlighting the potential of integrating GNNs and DRL as a flexible foundation for advanced warehouse optimization.

Keywords: order picking; deep reinforcement learning; graph neural networks; graph-based representation learning; warehouse routing optimization

1. Introduction

Order picking accounts for up to 55% of total warehouse operational costs, making it one of the most critical and labor-intensive processes in warehouse operations [1]. The order picking problem refers to minimizing the total distance travelled by a picker when visiting a set of storage locations to retrieve items for an order or group of orders. This problem can be viewed as a variation of the well-known Traveling Salesman Problem (TSP) and is NP-hard. Moreover, many related warehouse optimization problems such as order batching and storage location assignment are closely connected to order picking. While a wide range of heuristics and optimization methods have been proposed, most approaches are designed for individual subproblems and operate offline, without supporting real-time decision making [1].

The TSP is typically formulated as a graph-based problem. Graphs can be represented in two common ways: 1. where nodes correspond to coordinates in a geometric space, and 2. where nodes are defined by pairwise distances in a completely weighted graph. Warehouse layouts often impose grid-like constraints (orthogonal aisles and cross-aisles) [1], where distances follow Manhattan metrics rather than direct Euclidean distances. To model these settings, warehouse graphs are usually constructed with intersections and storage locations as nodes, with edges representing feasible traversal paths. These graphs are reduced to or constructed only from storage locations on order path, with distances computed via shortest paths through the layout [2].

 doi:10.20944/preprints202602.0896.v1

Recent advances in Deep Reinforcement Learning (DRL) have shown promising results for combinatorial optimization tasks such as the TSP by training agents to take actions based on environment states and rewards from reward functions [3]. Although a few studies have explored DRL in warehouse optimization, it remains limited and underexplored [4–6]. Use of DRL in warehouse environments could enable agents capable of making dynamic operational decisions, such as route selection, order batching, and item allocation.

A promising research direction lies also in graph representation learning, which aims to learn vector embeddings of graphs, nodes, or edges [7]. Since warehouses can be represented as graphs, Graph Neural Networks (GNNs) [8] can offer a powerful way to extract embeddings for storage locations and other entities. These embeddings could capture structural information about warehouse layouts and provide richer inputs for decision-making tasks compared to hand crafted features. For example, item allocation policies might be improved by leveraging learned embeddings that encode layout-specific characteristics.

In this work, we propose an integrated GNN-based deep reinforcement learning framework for warehouse optimization that jointly addresses order-picking path optimization and warehouse layout representation learning. Specifically, we aim to demonstrate the feasibility of training GNN models to produce near optimal order picking routes and learn embeddings of warehouse storage locations. Currently we focus only on optimizing order picking path, but we envision extending this approach to other problems by including orders and items as graph nodes and defining new reward functions, thereby laying the groundwork for a foundational GNN-DRL model that can be fine-tuned for diverse warehouse optimization tasks.

DRL applications with GNNs integration in domain of internal logistics remain scarce [9–11], and to the best of our knowledge, no existing study explicitly focuses on representation learning, leaving significant unexplored potential for advancing real-time, data-driven warehouse decision making.

Accordingly, this paper addresses the following research question: can graph neural networks learn meaningful embeddings of warehouse layouts that capture their structural properties while producing near-optimal order-picking paths, and can such models generalize across different warehouse configurations?

The main contributions of this paper are threefold:

(i) we formulate the warehouse order-picking problem as a graph-based deep reinforcement learning task using graph neural networks.

(ii) we demonstrate that the proposed GNN–DRL approach outperforms classical routing heuristics in simulated warehouse environments, with increasing advantages for larger and more complex orders and can generalize across different warehouse configurations with appropriate fine-tuning.

(iii) we show that the learned node embeddings capture meaningful structural properties of warehouse layouts.

We evaluate the proposed approach in simulated warehouse environment and discuss its applicability to a broader class of warehouse optimization problems. Overall, the presented GNN–DRL framework provides a unified methodology for order-picking optimization and warehouse representation learning, and establishes a flexible foundation for advanced, real-time, data-driven warehouse decision-making systems.

2. Materials and Methods

In this study we used synthetic warehouse layouts (for example see Figure 3, Figure 11, Figure 13) that were modelled as graphs, transforming them into inputs suitable for Graph Neural Networks (GNNs). GNN was trained using Deep Reinforcement Learning (DRL) to address the order-picking problem, with the additional objective of learning graph representations that capture warehouse structure and support decision-making. Our methodology follows a multi-step pipeline consisting of the following steps:

 doi:10.20944/preprints202602.0896.v1

1. Graph Construction from Relational Data: Warehouse layouts are modeled as graphs derived from relational database representations of storage locations and aisles of synthetic warehouse layouts. We define nodes corresponding to intersections and storage locations, and edges corresponding to feasible traversal paths. We employ both PyTorch Geometric [12–14] and NetworkX [15] python libraries to construct graphs. We use PyTorch graph for GNNs training and NetworkX for calculation of shortest distances for reward functions and for construction of distance matrices. PyTorch Geometric graphs are processed to conform to the input requirements of GNN training;
2. Graph Validation: To validate the correctness of the constructed graphs, we compute and visualize optimal picking paths using Dynamic programing [16]. This ensures that the graph representation accurately reflects warehouse layout constraints and distances;
3. Training GNN with DRL: We integrate GNNs with DRL to learn both embeddings of storage locations that capture layout structure and policy for generating optimized picking routes. The DRL agent produces actions based on warehouse graphs and receives rewards based on path efficiency;
4. Evaluation of order picking paths produced by GNN: We compare the performance of the GNN-DRL model against classical heuristics in terms of average travel distance and computational time. Algorithms used for comparison are: Dynamic programming [16], Christofides algorithm [17], , Local Search [18] and Lin-Kernighan heuristic [19]). We used python-tsp [20] implementation of those algorithms except for Christofides for which we used our own implementation;
5. Embedding Visualization and Evaluation: The learned embeddings from the GNN are visualized and evaluated to assess whether they capture meaningful structural properties of warehouse layouts.

All experiments are conducted on synthetic warehouse layouts generated to simulate realistic grid-like layouts. GNN training is run on a consumer-grade GPU (RTX 5060ti) inside of personal computer with 8 core CPU with 16 threads and 64GB of RAM, demonstrating the feasibility of our approach without requiring large-scale computational resource.

The warehouse layout is formalized as a weighted graph $G = (V, E)$, where the vertex (nodes) set V represents storage locations, aisles, intersections, and entrances, and edge weights E correspond to physical travel distances. Distances are derived from typical Warehouse Management System (WMS) data or, when unavailable, estimated from the geometric layout [21]. Edge weights (distances) between adjacent storage nodes are defined by their physical widths, while edges between stacked nodes reflect their stacking heights and connect exclusively to the corresponding ground-level location. Horizontally storage locations are connected to neighboring ground-level storage nodes, with additional edges introduced between nodes on opposite sides of an aisle and weighted by the aisle width. Edges connecting aisle nodes to storage nodes are assigned distances of full aisle length. Intersection nodes are positioned along aisles and connected to the intersecting aisles, the nearest storage nodes, and to successive intersections within the same aisle, thereby forming continuous traversal paths. Both intersection and aisle nodes are removed in the final graph representation for GNN training, after which storage locations previously connected through intersections are linked directly.

The demo warehouse layout used for training contains 600 storage locations, arranged along 3 vertical aisles and 10 horizontal aisles with storage locationg being assigned only to horizontal aisles, with two opposing storage walls per aisle of size x × y (x = 10, y = 3). In addition, the layout includes 15 intersections, modeled as graph nodes that connect vertical and horizontal aisles.

Figure 1 illustrates the database schema employed to construct the experimental warehouse graph. The schema formalizes the warehouse layout through relational entities representing warehouses, aisles, intersections, storage locations, entrance nodes, and edges. These entities and their relationships are used to represent the corresponding graph structure, where nodes represent

 doi:10.20944/preprints202602.0896.v1

physical locations and edges encode navigable connections. For graph construction, distances between intersections located on the same aisle are approximated as 3 meters.

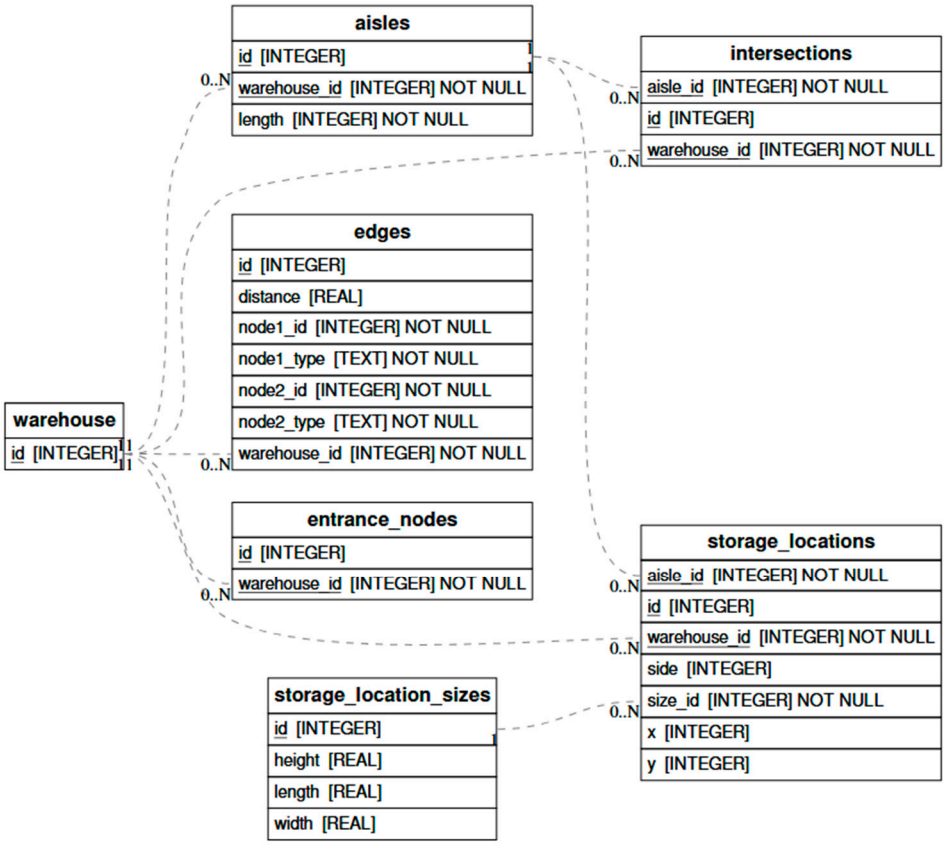


Figure 1. Diagram of Database Schema for Experimental Warehouse Graph Generation.

Figure 2 depicts the graph-based representation of the experimental warehouse layout (see Figure 3, Figure 11, Figure 13) constructed using the Python NetworkX library. Nodes represent physical warehouse entities, including entrances, aisles, intersections, and storage locations, and are distinguished by color. Aisles 0, 6 and 12 are vertical aisles with no assigned locations. Edges denote feasible traversal paths between nodes. Similar graph structure as depicted in Figure 2 serves as the input to the GNN, supporting the learning of node embeddings and the extraction of warehouse structural representations for downstream decision-making in order-picking tasks.

 doi:10.20944/preprints202602.0896.v1

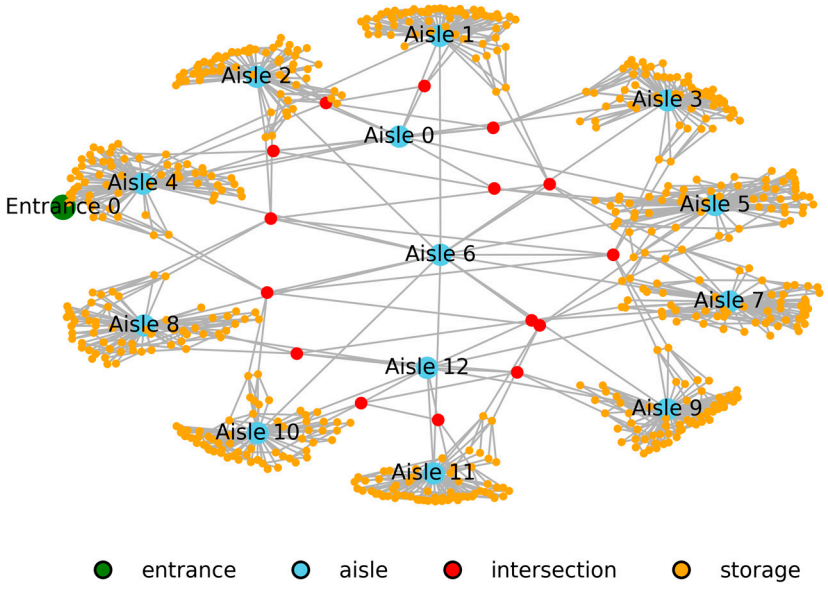


Figure 2. Graph-Based Representation of the Warehouse Layout for GNN-Based Order Picking.

Figure 3 illustrates an example of an optimal order-picking route generated from a graph-based representation of the warehouse in a top-down view. The route is used to validate the constructed warehouse graph model. The warehouse entrance (denoted by E) represents the starting and ending location of the picking tour. Storage locations with visible ids and marked with green are ones on the picking path. The numbers displayed above the blue points denote the order in which locations are visited in the optimal picking sequence. Shortest-path distances between all pairs of storage locations are computed using Dijkstra's algorithm [22], while the overall optimal picking sequence is determined using dynamic programming.

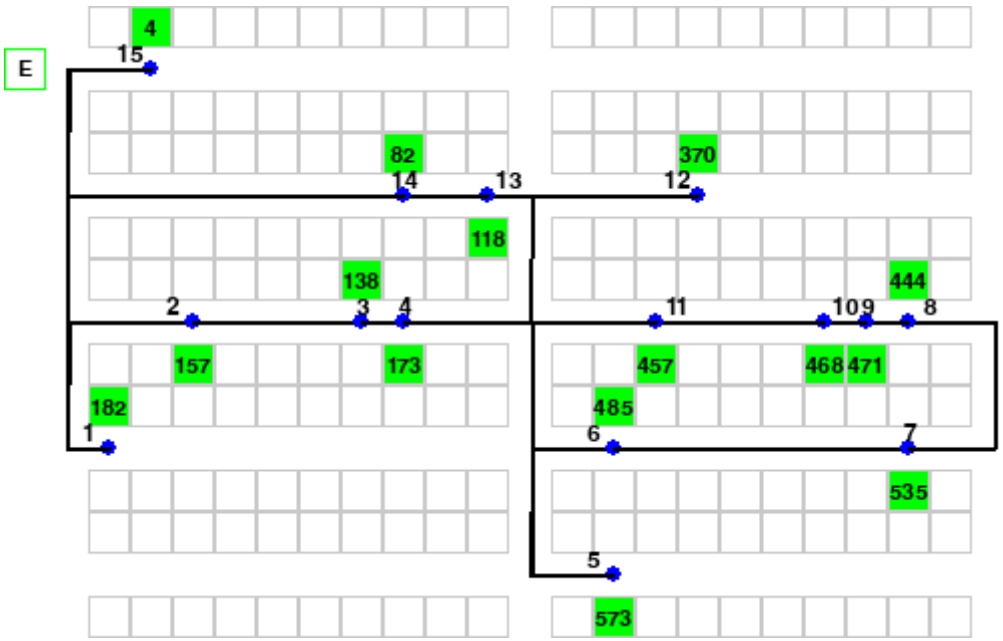


Figure 3. Visualization of an Optimal Order-Picking Path Derived from the Warehouse Graph Model.

The diagram in Figure 4 illustrates the process of constructing the initial graph structure from warehouse layout data. It begins with retrieving data from a database and then creating nodes

 doi:10.20944/preprints202602.0896.v1

representing storage locations, aisles, intersections, and the entrance. Edges are added between nodes to reflect connectivity, with distance attributes based on physical measurements (such as storage width or height, and aisle length or width).

Next, all pairs shortest-path distances are computed using Dijkstra's algorithm. For GNN processing, intersection and aisle nodes are subsequently removed, and storage nodes that were previously connected through these nodes are directly connected. This graph simplification was observed to have a positive impact on the quality of storage-node embedding representations, while no change, in the average shortest-path distance in early training stages was observed, when compared to the original graph containing intersection and aisle nodes. We attribute this to the aggregation behavior of GNN layers, whereby storage nodes adjacent to any intersections become more similar to one another.

Finally, a set of anchor nodes (e.g. the first, middle, and last node id) is selected, and additional anchor-type edges are introduced to store shortest-path distances from each node to these anchor points. All distances are then normalized using minimum and maximum distances from anchor edges, resulting in a graph representation that is subsequently used for training and simulation.

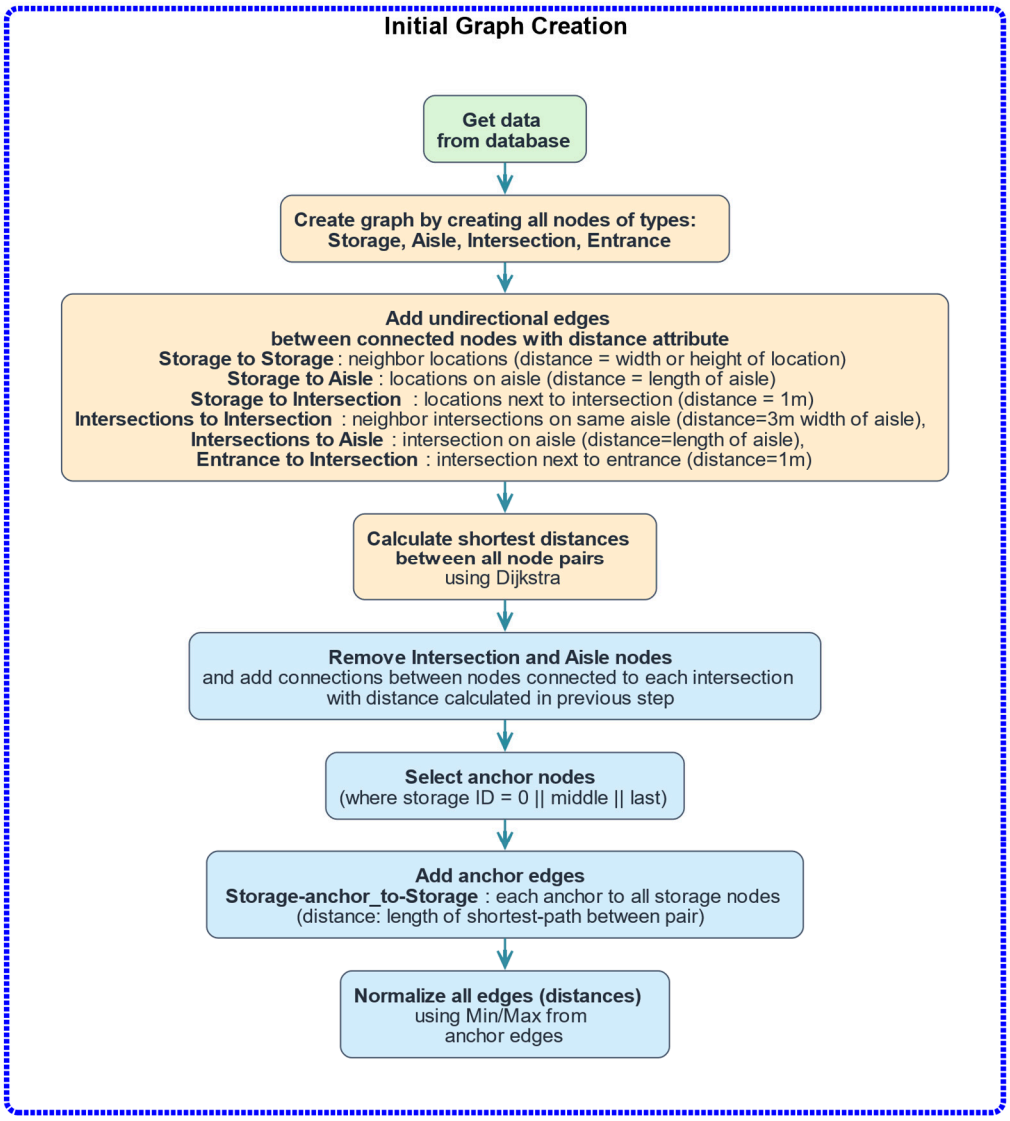


Figure 4. Graph Construction and Preprocessing Pipeline for the Warehouse Layout (blue steps are only done for GNN processing).

The GNN is trained using Proximal Policy Optimization (PPO) [23], which is an improvement over policy gradient methods [24]. The algorithm aims to maximize the average of action probabilities

 doi:10.20944/preprints202602.0896.v1

of taken actions based on the advantage estimator. Actions with higher advantage are reinforced, increasing their probability in future decisions, whereas actions with lower or negative advantage reduce their likelihood of being chosen in future decisions. This guides the policy toward selecting more effective actions and gradually improving overall performance. The policy is optimized by maximizing the policy gradient objective using gradient ascent, which is equivalent to minimizing the negative objective via gradient descent. Formally, the policy gradient loss is defined as:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{1}$$

where π_θ denotes the policy parameterized by θ, representing the probability distribution over possible actions. $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t given the state s_t at time step t and $\hat{A}_t$ is the estimated advantage function, which measures how much better (or worse) an action is compared to the expected value under the current policy. $\hat{\mathbb{E}}_t$ represents empirical expectation over time steps, approximated in practice by averaging over a batch.

In algorithms like PPO, we are not just increasing the raw probability of an action in isolation; we are increasing the probability relative to the previous policy. Algorithms control updates relative to the previous policy rather than boosting absolute probabilities. To quantify this change, PPO defines a probability ratio between the new and old policies as

$$R_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2}$$

which measures how much more (or less) likely the current policy π_θ is to select action a_t in state s_t compared to the previous policy $\pi_{\theta_{old}}$. The unclipped objective is then expressed as

$$L(\theta) = \hat{\mathbb{E}}_t\left[\mathrm{R}_t(\theta)\,\hat{A}_t\right] \tag{3}$$

In PPO, the clipping mechanism is used to prevent the policy from changing too much in a single update. While the policy ratio measures how much more (or less) likely the new policy is to select an action compared to the old policy, large deviations can destabilize learning. Clipping restricts the ratio to stay within a small range (e.g., $1 \pm \epsilon$). If the ratio moves outside this range, the objective uses the clipped value instead. This ensures that actions with high advantage are still encouraged, but the update remains controlled, preventing overly aggressive policy changes that could harm performance. The clipped objective is defined as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(\mathrm{R}_t(\theta)\,\hat{A}_t, clip(R_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{4}$$

where ϵ is the clipping parameter (typically set to 0.1 – 0.2).

Full PPO loss function is sum of L^{CLIP} and value loss L^{Value} that is mean squared error between predicted and target value estimation multiplied by value coefficient and subtracted by entropy of probability distribution multiplied by entropy coefficient). The full loss is defined as:

$$L = L^{CLIP}(\theta) - c_{value}L^{Value}(\theta) + c_{entropy}\mathcal{H}(\pi_\theta) \tag{5}$$

where c_{value} and $c_{entropy}$ are weighting coefficients for the value loss and entropy. Increasing the entropy of the policy helps promote exploration during training. A higher-entropy policy corresponds to a more diverse or less deterministic action distribution.

To improve training stability, methods such as Generalized Advantage Estimation (GAE) [25] are used. Without GAE, when the full sequence of rewards for an episode is available, the value target for the critic can be computed as the discounted sum of all future rewards. In our setting, the reward corresponds to the negative distance to the next location, and at the final timestep of an episode (when the agent reaches the last location in the path), we add the distance back to the entrance. All distances are normalized using the minimum and maximum values derived from anchor distances. Formally, target value estimate $\hat{V}_t$ is defined as:

$$\hat{\mathrm{V}}_t = \sum_{l=0}^{\infty} \gamma r_{t+1} \tag{6}$$

 doi:10.20944/preprints202602.0896.v1

where r_{t+1} is the reward at timestep t and γ is the discount factor. The corresponding advantage can then be expressed as:

$$A_t = \hat{V}_t - V(s_t) \tag{7}$$

where $V(s_t)$ denotes the value function prediction for state (s_t) and $\hat{V}_t$ is target value estimate.

GAE improves this by using a smoothed combination of temporal-difference errors denoted as δ_t. across multiple time steps. The temporal-difference at time step t is defined as:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{8}$$

where r_t is the immediate reward, $V(s_t)$ and $V(s_{t+1})$ are value function estimates for the current and next state, respectively, and γ is the discount factor. The parameter λ is subsequently used in GAE to control the exponential weighting of δ_t over time. Advantage estimation $\hat{A}_t$ is then calculated as sum of δ_t and advantage of next step. $\hat{A}_t$ satisfies the following recurrence relation:

$$\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1} \tag{9}$$

where λ is the GAE smoothing parameter. The value target $\hat{V}_t$ is obtained by adding the advantage estimate $\hat{A}_t$ to the baseline value function estimate $V(s_t)$.

$$\hat{V}_t = \hat{A}_t + V(s_t) \tag{10}$$

We applied Generalized Advantage Estimation (GAE) using a smoothing parameter $\lambda = 0.9$ and a discount factor $\gamma = 0.99$.

The diagram (Figure 5) illustrates the workflow of a single episode during simulation and training of the GNN-based order-picking agent. Each episode begins with the generation of random orders, after which all subsequent steps are executed in parallel for the entire batch of orders. For each order, a copy of the initial warehouse graph is constructed, and additional features are assigned to storage nodes to indicate their role. Storage nodes along the current path are marked with a binary feature (0 or 1) indicating whether they are part of the active path. An additional picker node is introduced to represent the current location of the picker. This node is connected to the entrance node or storage node that the picker most recently visited. GNN produces two outputs:

- A score for each storage node, representing the likelihood that the node should be chosen as the next step in the path.
- A graph-level score, obtained via mean pooling over all storage node representations in the final GNN layer followed by feed forward layer (linear), giving value estimation (approximation of total distance left on order picking path).

During action selection, non-candidate storage nodes (those not on the path) are masked out, and a SoftMax is applied over candidates scores to generate probabilities. During training, the next action is sampled from this distribution, while during testing, the action with the maximum probability (argmax) is chosen. During training we remove last step from each order from our training dataset as number of candidates is only 1.

The resulting states, actions, value estimates, and rewards are stored, and the process is repeated until all required storage locations for each order have been visited. After completion of the picking tour, rewards are computed based on the total travel distance, including the return to the entrance, and normalized using the minimum and maximum values derived from anchor-based distances. Finally, the GNN is trained for several epochs using the collected episode data, after which the system proceeds to the next episode and repeats the cycle.

 doi:10.20944/preprints202602.0896.v1

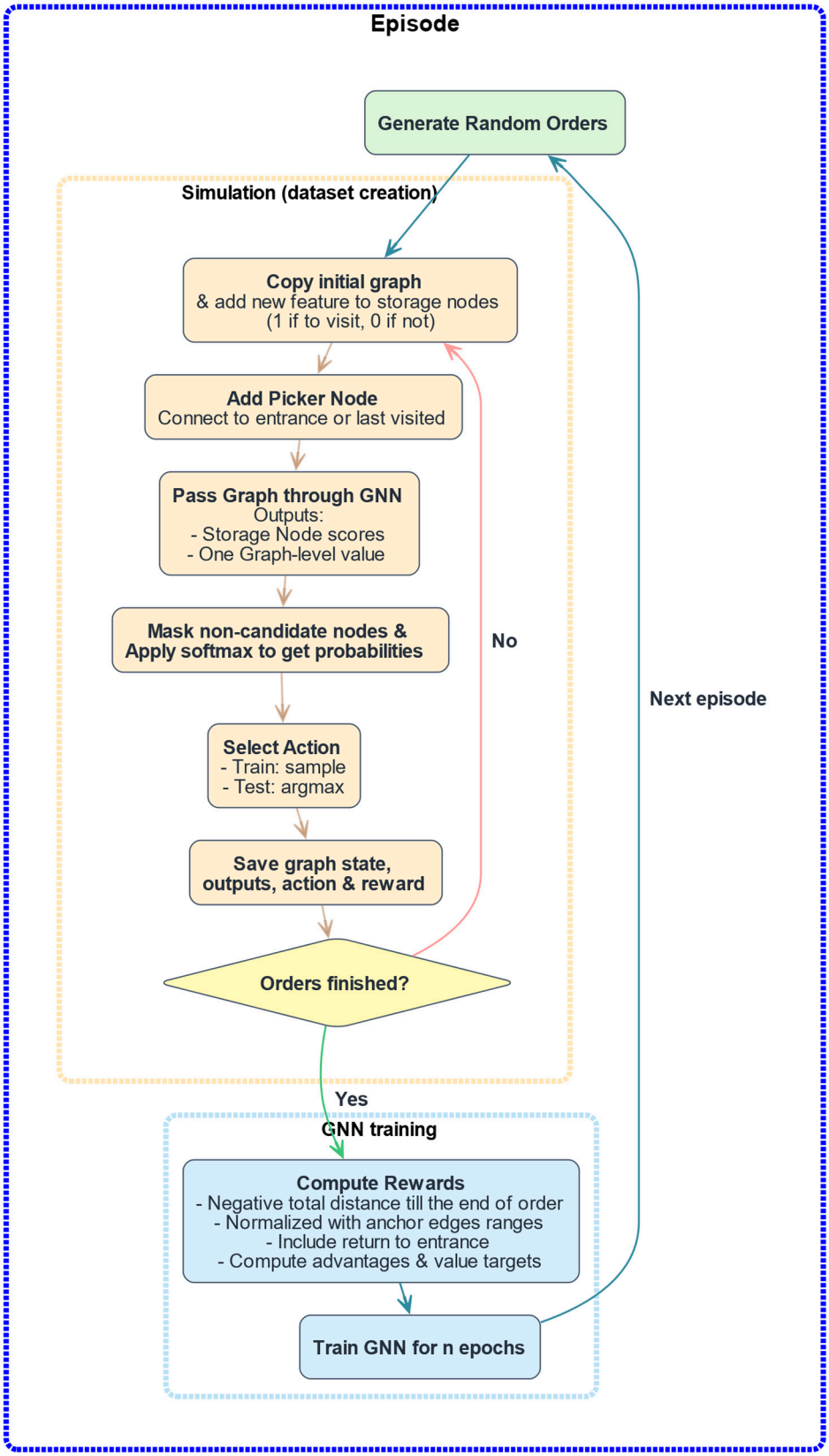


Figure 5. Episode Workflow for Simulation and Training of the GNN-Based Order-Picking Agent.

GNN training hyperparameters are set as follows: the entropy coefficient is 0.001, the value function coefficient is 0.5, and the PPO clipping parameter (clip_eps) is 0.1. Optimization is performed using the AdamW [26] optimizer with a learning rate of 1×10^{-4} and a batch size of 64.

 doi:10.20944/preprints202602.0896.v1

At each episode, 42 random orders of size 25 are generated. For each order, the picker's path is simulated, rewards are computed, and the model is trained for 8 epochs using the collected trajectories.

The model architecture begins with an embedding layer which increases dimensionality to 32. This is followed by 12 TransformerConv [27] layers with 4 attention heads and an output dimension of 32. After the TransformerConv layers, a linear layer is applied to nodes of the target type (storage nodes), followed by an additional linear layer that produces node-level scores. A SoftMax function is then applied to obtain action probabilities.

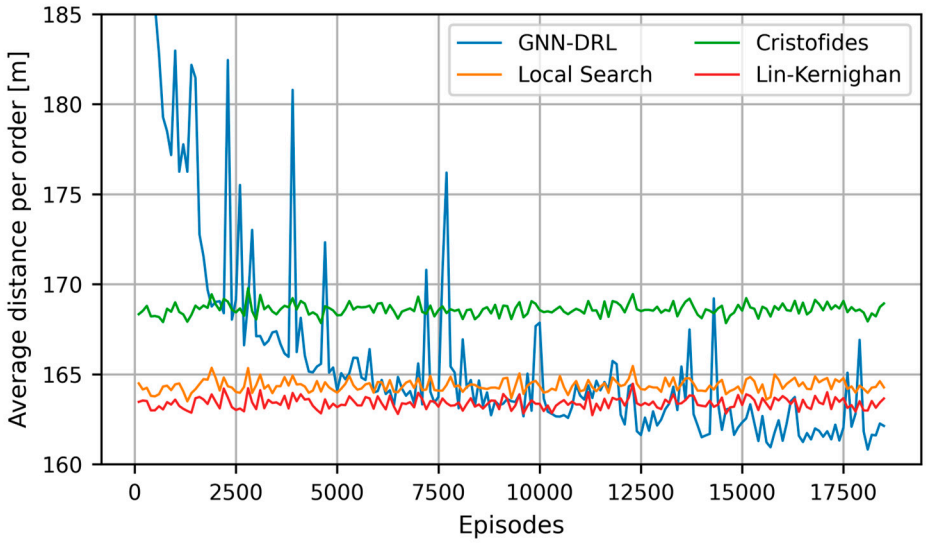
To estimate the global value, mean pooling is applied to the node embeddings after the first linear layer, followed by a linear layer that outputs the graph-level value estimate.

3. Results

Figure 6 shows the progression of the average distance per order during training of the GNN-DRL model and its comparison with classical heuristics. The x-axis represents the number of training episodes, ranging from 0 to 18,500, while the y-axis denotes the average distance traveled per order, measured in meters. Performance is evaluated every 100 episodes on new set of 1,000 randomly generated orders of size 25. The GNN-DRL model is compared against three baseline methods: Local Search, Christofides, and Lin–Kernighanas indicated in the legend. To make figure more readable y-axis is limited to value of 185 m (values for episodes 100, 200, 300, 400 with y values cut from figure are approximately 259 m, 211 m, 189 m, 189 m).

During early training, the GNN-DRL model exhibits higher variance and longer average distances, followed by a gradual improvement as training progresses. After approximately 3,000 episodes, the GNN-DRL model consistently outperforms the Christofides heuristic; after around 8,000 episodes, it surpasses Local Search; and after approximately 12,000 episodes, it achieves sustained improvements over the Lin–Kernighan heuristic. After that GNN-DRL model maintains a consistently lower average distance per order than Local Search, Christofides, and Lin–Kernighan, demonstrating its advantage in the converged regime.

Temporary spikes in the GNN-DRL performance curve are visible throughout training and are attributed to continued exploration, which may allow the model to escape local minima. Each training episode requires on average 1.34 minutes, resulting in a total training time of approximately 17 days for 18,500 episodes of continuous training. Note that this is one time cost and inference is done in around 0.057 seconds for every step (Simulation part in Figure 5) so for order of size 15 takes around 0.86 seconds and for order of size 25 it takes around 1.44 seconds.



 doi:10.20944/preprints202602.0896.v1

Figure 6. Training Convergence of the GNN-DRL Model Compared to Classical Heuristics.

In the final phase of training, the model was trained on orders with sizes ranging from 5 to 25 storage locations, using 42 randomly generated orders per order size for each episode for a total of 50 training episodes. The trained model was subsequently evaluated on 1,000 randomly generated orders for each order size between 5 and 35.

Figure 9 jointly illustrates the effect size and statistical significance of the performance difference between the GNN-DRL model and the Lin–Kernighan heuristic as a function of order size. Statistical significance is assessed using a Wilcoxon signed-rank test [28] on the route lengths of 1,000 orders per order size. The x-axis represents the order size, while the left y-axis shows the relative improvement of GNN-DRL over Lin–Kernighan, expressed as the percentage reduction in average route length. Negative values indicate superior performance of the GNN-DRL model. The right y-axis reports the statistical significance of this difference using the Wilcoxon signed-rank test, expressed as $-\log_{10}(p\text{-value})$, with the dotted horizontal line indicating the conventional significance threshold of $p = 0.05$.

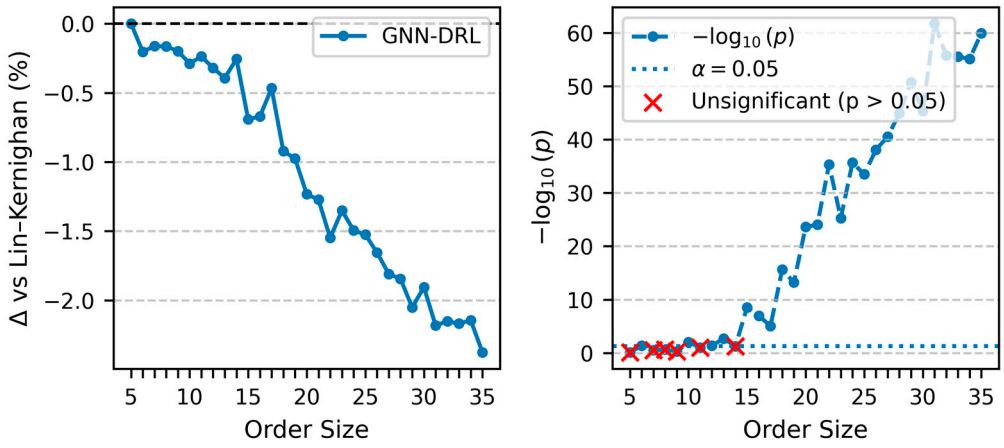


Figure 7. Relative performance improvement of GNN-DRL compared to Lin–Kernighan (left) and the corresponding statistical significance (right) across different order sizes for the smaller warehouse instance.

Statistically significant improvements (p-value < 0.05) emerge from order size 15 onward, with significance increasing rapidly for larger orders and reaching extremely small p-values (below 10^{-30}) for most order sizes above 20. The standard deviation values for both methods remain comparable across all order sizes, indicating that the observed performance gains are not driven by increased variance or outliers.

The corresponding Table 1 provides a quantitative comparison between GNN-DRL and Lin–Kernighan across all evaluated order sizes, reporting average route lengths, percentage changes, standard deviations, and Wilcoxon signed-rank test statistics. For small order sizes (5–10 locations), average route lengths are nearly identical, and differences are not consistently statistically significant. As order size increases, the GNN-DRL model increasingly outperforms Lin–Kernighan, as reflected by a growing negative percentage change in route length.

Table 1. ✻omparison of GNN-DRL and Lin–Kernighan Performance Across Order ✻izes.

Orde ✻ize	Avg. lengths DNNDRL	Avg. lengths Lin–Kernighan	% change	✻tat Wilcoxon	p-value Wilcoxon	✻tandard deviation GNN	✻tandard deviation LK
5	72.29	72.29	0.00	13377	8.04E-01	9.36	9.46
6	79.08	79.24	-0.20	18518	4.52E-02	9.18	9.20
7	84.83	84.97	-0.16	27512	3.18E-01	9.06	9.03

 doi:10.20944/preprints202602.0896.v1

8	90.95	91.10	-0.16	38477.5	2.29E-01	9.03	8.93
9	96.07	96.26	-0.20	65057	5.26E-01	9.14	9.18
10	101.20	101.49	-0.29	61272	9.70E-03	9.58	9.36
11	106.45	106.71	-0.24	73991.5	9.33E-02	9.83	9.67
12	111.17	111.53	-0.32	95931.5	4.52E-02	9.66	9.61
13	114.87	115.33	-0.39	95396.5	1.91E-03	9.81	9.68
14	119.95	120.26	-0.26	115454.5	5.58E-02	9.76	9.37
15	124.46	125.33	-0.69	99094	2.92E-09	9.80	9.83
16	128.91	129.78	-0.67	119242.5	1.05E-07	10.17	10.09
17	132.94	133.56	-0.47	121430.5	9.14E-06	9.86	10.03
18	136.52	137.79	-0.92	104934	2.41E-16	10.17	10.42
19	140.19	141.57	-0.98	116505	5.90E-14	9.74	9.99
20	143.87	145.66	-1.23	106291.5	2.35E-24	10.11	10.24
21	147.63	149.54	-1.27	108883	8.96E-25	9.63	9.82
22	150.93	153.30	-1.55	100548.5	4.72E-36	9.78	10.07
23	154.21	156.32	-1.35	110766.5	5.27E-26	9.55	10.07
24	157.76	160.15	-1.49	98666	2.12E-36	9.37	9.92
25	160.77	163.26	-1.53	105693	3.29E-34	9.54	9.99
26	163.91	166.67	-1.66	105217.5	8.33E-39	9.63	10.09
27	166.88	169.96	-1.81	102923.5	2.88E-41	10.10	10.22
28	169.32	172.50	-1.84	96287.5	1.24E-45	9.96	10.08
29	173.10	176.72	-2.05	86931.5	1.84E-51	9.60	10.21
30	176.08	179.50	-1.90	99242	4.27E-46	9.75	10.17
31	178.68	182.66	-2.18	80420.5	1.87E-62	9.10	9.96
32	181.63	185.62	-2.15	87307	1.66E-56	9.50	10.37
33	183.67	187.74	-2.17	90216	2.99E-56	9.65	9.92
34	186.87	190.97	-2.14	88614.5	7.82E-56	9.77	10.41
35	189.10	193.70	-2.38	86958	1.11E-60	9.87	10.76

Overall, the results shown in Figure 7 and the accompanying table (Table 1) demonstrate that the performance advantage of the GNN-DRL approach becomes more pronounced as order size and combinatorial complexity increase, confirming its effectiveness for larger and more challenging order-picking problems.

When the model trained on the smaller warehouse was directly applied to the larger warehouse containing approximately 1,200 storage locations, a decrease in performance was observed. To address this, the model was fine-tuned on the larger warehouse by training for an additional 2,800 episodes using orders of size 25, with 4 training epochs per episode. Due to GPU memory constraints, the batch size was reduced from 64 (used for the smaller warehouse) to 48, and the learning rate was decreased from 1×10^{-4} to 1×10^{-5}. After this fine-tuning phase, an additional 50 episodes of training were performed using variable order sizes ranging from 5 to 25 locations, following the same procedure as for the smaller warehouse.

Figure 8 illustrates the performance of the GNN-DRL model during fine tuning and baseline heuristics when applied to a larger warehouse instance. The x-axis shows the number of training episodes, while the y-axis reports the average distance per order in meters. Results are shown for the GNN-DRL model and three comparison methods: Local Search, Christofides, and Lin–Kernighan.

As shown in Figure 8, the GNN-DRL model consistently achieves lower average distances per order than the Lin–Kernighan heuristic throughout training, despite the increased problem scale. However, the relative performance improvement is smaller than that observed for the smaller warehouse, indicating a slight reduction in percentage gain as warehouse size increases. These results demonstrate that while the GNN-DRL approach generalizes to larger warehouse layouts, additional fine-tuning is necessary and performance gains diminish moderately with increase of total number of storage locations in warehouse.

 doi:10.20944/preprints202602.0896.v1

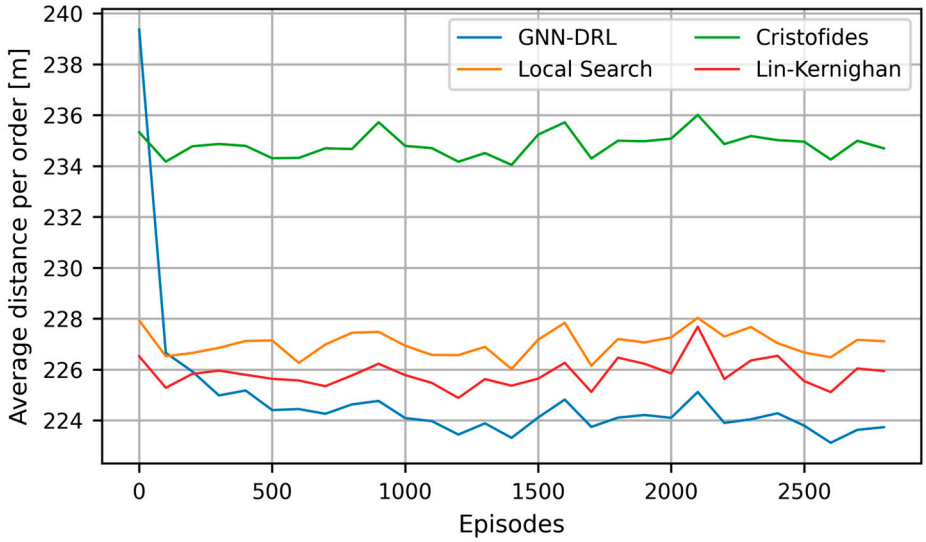


Figure 8. GNN-DRL Model After Fine-Tuning on a Larger Warehouse Instance.

Figure 9 illustrates the effect size and statistical significance of the performance difference like Figure 7 but for bigger warehouse instance. The solid curve with circular markers represents the percentage improvement of GNN-DRL relative to Lin–Kernighan. For small order sizes, improvements are close to zero and fluctuate around the baseline, indicating negligible performance differences. Order sizes for which the Wilcoxon test does not indicate statistical significance ($p \geq 0.05$) sizes 7, 8, 9, and 11 are highlighted with large red cross markers. These cases correspond to the entries in the table (Table 2) where p-values exceed the 0.05 threshold, confirming that observed differences for small order sizes are not statistically significant.

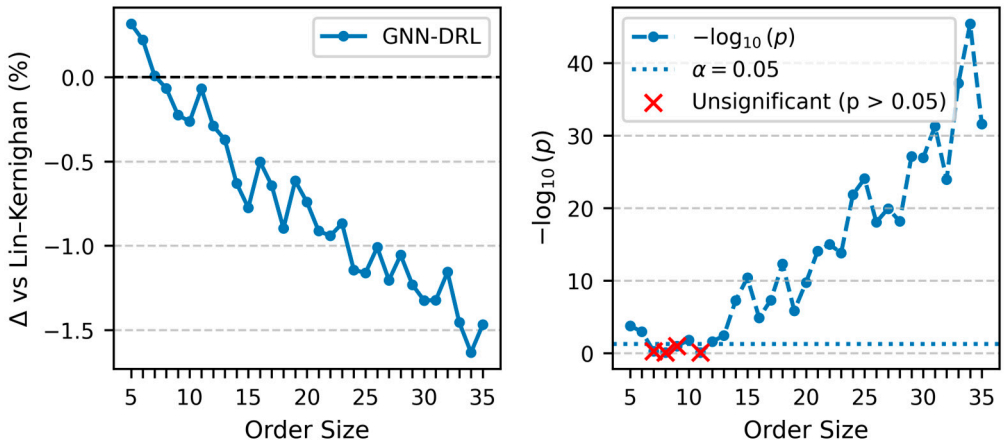


Figure 9. Relative performance improvement of GNN-DRL compared to Lin–Kernighan (left) and the corresponding statistical significance (right) across different order sizes for the larger warehouse instance.

As order size increases, the magnitude of improvement steadily grows, reaching reductions of approximately 1.0–1.6% for larger orders. In parallel, the $-\log_{10}(p\text{-value})$ curve rises sharply, indicating rapidly increasing statistical significance. From order size 11 onward, all improvements are statistically significant, with p-values decreasing by several orders of magnitude. For orders above 24 locations, the p-values reported in the table mostly fall below 10^{-20}, demonstrating that the observed performance gains are both substantial and highly robust.

 doi:10.20944/preprints202602.0896.v1

Table 2. Average Route Lengths and Wilcoxon Signed-Rank Test Results for GNN-DRL and Lin–Kernighan Across Order Sizes.

Orde ⚜ize	Avg. lengths DNNDRL	Avg. lengths Lin–Kernighan	% change	⚜tat Wilcoxon	p-value Wilcoxon	⚜tandard deviation GNN	⚜tandard deviation LK
5	98.51	98.20	0.31	16723.0	1.77E-04	12.01	11.99
6	107.38	107.14	0.22	24573.0	1.11E-03	11.57	11.66
7	115.48	115.47	0.01	40093.0	5.47E-01	11.57	11.59
8	123.34	123.42	-0.07	65723.5	8.32E-01	11.29	11.46
9	130.20	130.50	-0.23	66902.0	1.03E-01	12.38	12.43
10	137.74	138.11	-0.26	80217.5	1.57E-02	11.84	11.69
11	143.94	144.04	-0.07	92446.0	8.11E-01	11.92	11.98
12	150.09	150.52	-0.29	96304.0	2.50E-02	12.11	11.99
13	156.91	157.50	-0.37	98883.0	3.89E-03	11.84	11.97
14	163.46	164.50	-0.63	98354.0	5.49E-08	12.13	12.46
15	169.52	170.84	-0.77	95319.5	4.16E-11	12.32	12.52
16	175.80	176.69	-0.50	121590.5	1.32E-05	12.65	12.86
17	181.67	182.85	-0.64	120991.0	4.97E-08	13.21	13.31
18	187.31	189.00	-0.90	119723.0	5.11E-13	12.92	13.32
19	192.95	194.15	-0.62	138789.0	1.39E-06	12.89	13.32
20	197.29	198.77	-0.74	132526.5	1.83E-10	13.30	13.93
21	202.95	204.82	-0.91	130349.0	8.72E-15	12.60	13.10
22	207.94	209.92	-0.94	129352.0	1.01E-15	13.41	13.68
23	213.94	215.81	-0.87	136634.0	1.57E-14	13.62	14.29
24	218.28	220.81	-1.14	124959.5	1.45E-22	13.53	14.24
25	223.75	226.37	-1.16	124855.0	8.71E-25	13.61	14.04
26	227.86	230.19	-1.01	134447.5	8.93E-19	13.44	14.14
27	233.15	235.99	-1.20	135423.0	1.18E-20	13.47	14.42
28	237.77	240.30	-1.05	131533.0	6.76E-19	14.03	14.66
29	240.56	243.56	-1.23	128496.5	7.21E-28	13.70	13.97
30	246.04	249.35	-1.32	122143.5	1.16E-27	13.41	14.26
31	250.68	254.04	-1.32	117603.0	5.30E-32	13.90	14.45
32	255.20	258.18	-1.16	137896.5	1.21E-24	13.65	14.31
33	259.02	262.85	-1.45	108961.0	6.04E-38	14.12	14.89
34	261.88	266.23	-1.63	102239.5	4.12E-46	13.86	14.75
35	267.86	271.85	-1.47	120201.5	2.70E-32	14.26	15.47

Overall, the Figure 9 and the accompanying Table 2 consistently show that while performance differences between GNN-DRL and Lin–Kernighan are marginal and statistically insignificant for small orders, the GNN-DRL model achieves increasingly larger and statistically significant improvements as order size and therefore combinatorial complexity increase. This confirms that the advantage of the GNN-DRL approach becomes more pronounced in larger and more challenging order-picking scenarios.

To better understand how the GNN captures the warehouse structure, we visualize the learned embeddings of storage nodes. The embeddings are taken from the output of the final TransformerConv layer under the conditions where all storage locations are marked as being on the order path and the picker location is set to the entrance. For visualization purposes, the embedding dimensionality is reduced to two using Uniform Manifold Approximation and Projection (UMAP) [29].

Figure 10 presents the resulting visualization alongside the corresponding warehouse layout grid. On the left, storage locations are colored according to their physical aisles, while on the right, colors represent k-means clusters [30] derived from the learned embeddings. The number of clusters

 doi:10.20944/preprints202602.0896.v1

is set equal to the number of aisles containing storage locations. Each point corresponds to a storage location: the point shape (square vs. circle) indicates the side of the aisle, border thickness encodes the x-coordinate (thicker borders correspond to larger x values), and opacity represents the y-coordinate, with more opaque points indicating higher vertical positions.

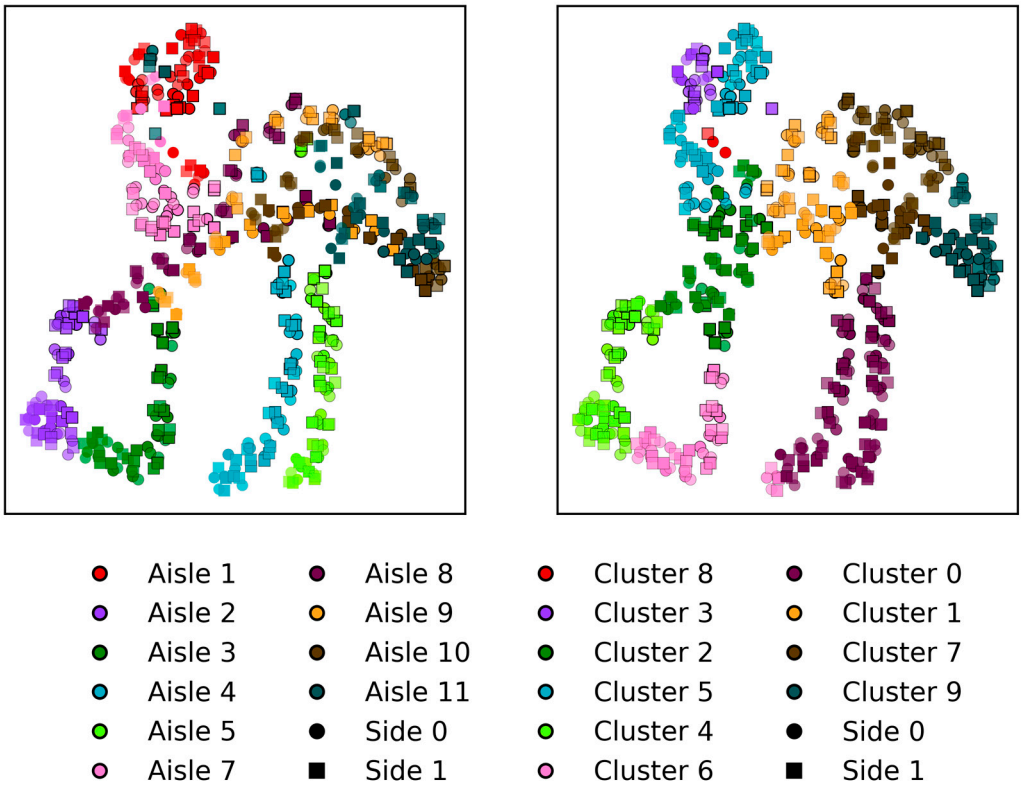


Figure 10. UMAP Visualization of Learned Storage-Node Embeddings Colored by Aisle (left) or K-means cluster (right) with picker location set to Entrance.

Spatial mapping enables a direct comparison between the learned embedding space and the physical warehouse layout. In the left part of Figure 10, colors denote individual aisles. While storage locations from the same aisle tend to form broadly coherent clusters, the embedding structure indicates that physical proximity, both between storage locations and relative to the warehouse entrance, also plays an important role and in some cases appears more influential than aisle membership. Storage locations that are spatially close to one another, as well as those with similar distances to the entrance, appear closer in the embedding space even when they belong to different aisles. The strong alignment between the relative positions of storage locations in the embedding space and their positions in the physical warehouse layout demonstrates that the learned embeddings successfully preserve meaningful spatial organization.

Figure 11 shows the corresponding warehouse layout grid, where storage locations are colored to match the k-means clusters visible in the UMAP visualization. Color is based on cluster of ground (y = 0) location. This spatial mapping allows a direct comparison between the learned embedding structure and the physical warehouse layout. The strong correspondence between UMAP projection, physical aisles and distance to entrance confirms that the learned embeddings preserve meaningful spatial organization.

Anchor nodes (with identifiers 0, 300, and 599) are visually distinguishable in the embedding space, indicating that anchor-based distance features influence the learned representations. While these anchors improve structural awareness during learning, it remains an open question whether their explicit inclusion could be reduced or removed without degrading model performance.

 doi:10.20944/preprints202602.0896.v1

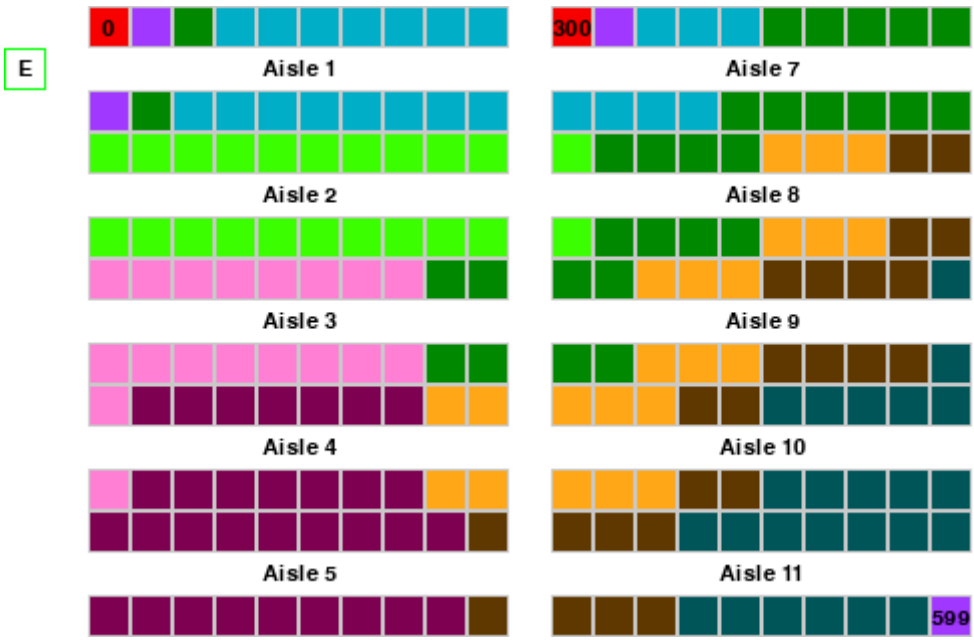


Figure 11. Mapping of k-Means Embedding Clusters to the Physical Warehouse Layout.

Figure 12 illustrates how the storage-node embeddings change when the picker's location is modified. In this experiment, the picker node is placed at storage location with id 495. Projection shows a noticeable reorganization of clusters, which visually correlates strongly with graph distance to the selected picker location. Notably, all locations within the same aisle as picker (Aisle 10) form a compact cluster in the embedding space.

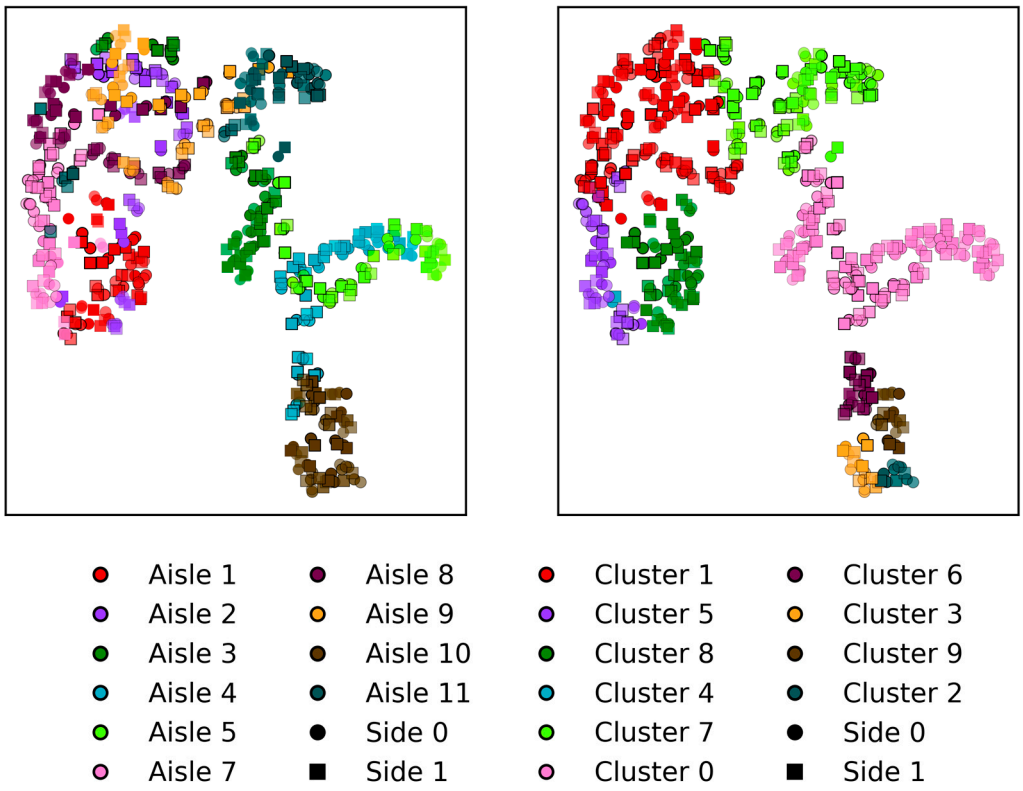


Figure 12. UMAP Visualization of Learned Storage-Node Embeddings Colored by Aisle (left) or K-means cluster (right) with picker location set to storage location in Aisle 10.

 doi:10.20944/preprints202602.0896.v1

Storage locations closer to the picker in the warehouse graph appear more similar in the embedding space which can be also identified in k-means clusters projected on warehouse layout (Figure 13). This behavior suggests that the learned embeddings dynamically encode task-relevant context, such as the current picker position. In practical applications, this property could be exploited to identify similar storage locations for items that frequently co-occur in orders, as well as to identify dissimilar locations for items that should be spatially separated (e.g., due to congestion or incompatibility constraints).

Figure 13. Effect of Picker Location (set to storage node with id 495) on mapping of k-Means Clusters to the Physical Warehouse Layout.

Presented results support the main motivation outlined in the Introduction: that graph-based representation learning achieved with deep reinforcement learning can effectively capture warehouse layout structure. The visualizations demonstrate that the GNN learns meaningful embeddings.

Importantly, these embeddings are not only descriptive but also operationally useful. The previously demonstrated improvements over classical heuristics indicate that the learned representations can directly support decision-making in order picking. The sensitivity of embeddings to the picker's location further highlights their adaptability to dynamic operational contexts.

Overall, these results validate proposed approach and illustrate the potential of GNN-based warehouse representation learning done with deep reinforcement learning for real-time, adaptive warehouse optimization, as anticipated in the Introduction.

4. Discussion

This study demonstrates that the combined GNN–DRL approach consistently produces more efficient order-picking paths than traditional routing heuristics across different warehouse layouts and a wide range of order sizes, with performance gains becoming more pronounced for larger orders. In contrast to most existing approaches that focus exclusively on optimizing routing decisions, the proposed method jointly learns order-picking policies and structural representations of warehouse layouts, enabling it to exploit relational information such as aisle connectivity, relative distances, and layout-induced constraints.

While the absolute improvements in route length are modest for small orders, they are practically relevant in realistic warehouse operations, where even small percentage reductions can

 doi:10.20944/preprints202602.0896.v1

accumulate into substantial savings at scale. The increasing advantage for larger orders reflects the model's improved performance as combinatorial complexity grows, with the GNN–DRL approach consistently achieving shorter routes than local heuristic strategies in these more challenging scenarios. At the same time, the results show that the learned embeddings consistently encode meaningful structural properties of the warehouse graph.

The embedding analysis together with clear improvements over traditional methods in order picking path lengths provides clear evidence that GNNs can capture meaningful structural information. UMAP visualizations reveal that storage locations belonging to the same aisle are visually distinguishable in embedding projections, while neighboring aisles are positioned close to each other in the embedding space. This indicates that warehouse structure is reflected in the learned representations. These observations confirm that the GNNs can properly encode higher-level structural properties of the warehouse layout. Unlike other DRL routing approaches, the learned representations can be inspected and related to physical warehouse structure, revealing clear alignment with aisle organization, spatial proximity, and picker location. This transparency supports interpretability by enabling validation of whether the model encodes meaningful and operationally relevant information rather than spurious patterns. The consistency between embedding structure, warehouse layout, and observed routing performance also serves as an additional sanity check for the correctness of the modeling assumptions and experimental setup, strengthening confidence in the validity of the proposed framework.

Further insight is provided by experiments in which the picker's location is changed. The resulting reorganization of the embedding space correlates strongly with graph distance to the picker, demonstrating that the learned representations are context-sensitive and dynamically adapt to task-relevant information. This property suggests that the embeddings could support additional warehouse decision-making tasks beyond routing, such as identifying similar or dissimilar storage locations based on operational criteria.

Regarding generalization, the results show that models trained on one warehouse layout can be transferred to larger layouts through additional fine-tuning. Although direct application without adaptation results in reduced performance, a limited fine-tuning phase is sufficient to restore and maintain a consistent advantage over classical heuristics. While relative improvements are slightly smaller in the larger warehouse, the GNN–DRL approach remains robust across scales, suggesting that the learned representations capture reusable, layout-agnostic structural patterns rather than overfitting to a single configuration.

Further research direction could be extending warehouse graph to include more graph entities like orders, items and more than one agent. This extension would allow joint optimization of tasks such as order batching and item allocation or reallocation, while extending representation learning to additional entities including orders and items.

Despite the encouraging results, several limitations of this study should be acknowledged. First, all experiments were conducted on synthetic warehouse layouts, which, while designed to reflect realistic grid-based structures, may not fully capture the heterogeneity and operational constraints of real-world warehouses. Second, the proposed GNN–DRL approach requires a non-trivial training time, which may limit its applicability in settings where rapid deployment or frequent retraining is required, although this cost is incurred offline and inference remains efficient. Third, the current formulation relies on anchor-based distance features to support representation learning; while effective, the extent to which these engineered features are necessary warrants further investigation. Finally, the experimental setup considers a single-picker, single-tour scenario, and does not yet address multi-picker interactions, congestion effects, or dynamic order arrivals, which are important aspects of practical warehouse operations and represent natural directions for future work.

Overall, the findings provide affirmative answers to the research question posed in this study. First, the results demonstrate that GNNs can learn embeddings that capture meaningful structural properties of warehouse layouts while providing improvement over traditional methods in order-picking performance. Second, the learned models exhibit the ability to generalize across different

 doi:10.20944/preprints202602.0896.v1

warehouse configurations, with fine-tuning providing an effective mechanism for adapting to new layouts. Taken together, these findings position GNN-DRL not merely as a competitive routing heuristic, but as a general representation-learning framework for adaptive, graph-structured warehouse decision making.

Author ✙ontributions: Conceptualization, N.Č. and A.Š.; methodology, N.Č.; software, N.Č.; validation, N.Č. and A.Š.; formal analysis, N.Č.; investigation, N.Č.; resources, A.Š.; data curation, N.Č.; writing—original draft preparation, N.Č.; writing—review and editing, N.Č. and A.Š.; visualization, N.Č.; supervision, A.Š.; project administration, A.Š.; funding acquisition, A.Š. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Slovenian Research and Innovation Agency (research core funding No. P5-0018), and the Ministry of Higher Education, Science, and Innovation of the Republic of Slovenia as part of the Next Generation EU National Recovery and Resilience Plan (Grant No. 3330-22-3515; NOO No: C3330-22-953012),.

Data Availability ✙tatement: The original contributions presented in this study are included in the article/supplementary material. Further inquiries can be directed to the corresponding author.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT (version 5.2) for language editing and stylistic refinement of the text. The authors reviewed and edited the generated content and take full responsibility for the final content of this publication.

✙onflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DRL	Deep Reinforcement Learning
GNN	Graph Neural Network
GNN-DRL	Graph Neural Network–Deep Reinforcement Learning
TSP	Traveling Salesman Problem
UMAP	Uniform Manifold Approximation and Projection
PPO	Proximal Policy Optimization
GAE	Generalized Advantage Estimation
GPU	Graphics Processing Unit
WMS	Warehouse Management System
GPU	Graphics Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory

References

1. Masae, M.; Glock, C.H.; Grosse, E.H. Order Picker Routing in Warehouses: A Systematic Literature Review. *Int. J. Prod. Econ.* **2020**, *224*, 107564, doi:10.1016/j.ijpe.2019.107564.
2. Ratliff, H.D.; Rosenthal, A.S. Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. *Oper. Res.* **1983**, *31*, 507–521, doi:10.1287/opre.31.3.507.
3. Mazyavkina, N.; Sviridov, S.; Ivanov, S.; Burnaev, E. Reinforcement Learning for Combinatorial Optimization: A Survey. *Comput. Oper. Res.* **2021**, *134*, 105400, doi:10.1016/j.cor.2021.105400.
4. Cals, B.; Zhang, Y.; Dijkman, R.; van Dorst, C. Solving the Online Batching Problem Using Deep Reinforcement Learning. *Comput. Ind. Eng.* **2021**, *156*, 107221, doi:10.1016/j.cie.2021.107221.
5. Mahmoudinazlou, S.; Sobhanan, A.; Charkhgard, H.; Eshragh, A.; Dunn, G. Deep Reinforcement Learning for Dynamic Order Picking in Warehouse Operations. *Comput. Oper. Res.* **2025**, *182*, 107112, doi:10.1016/j.cor.2025.107112.

 doi:10.20944/preprints202602.0896.v1

6. Wang, X.; Zhang, L.; Wang, L.; Zuñiga, E.R.; Wang, X.V.; Flores-García, E. Dynamic Multi-Tour Order Picking in an Automotive-Part Warehouse Based on Attention-Aware Deep Reinforcement Learning. *Robot. Comput.-Integr. Manuf.* **2025**, *94*, 102959, doi:10.1016/j.rcim.2025.102959.
7. Ju, W.; Fang, Z.; Gu, Y.; Liu, Z.; Long, Q.; Qiao, Z.; Qin, Y.; Shen, J.; Sun, F.; Xiao, Z.; et al. A Comprehensive Survey on Deep Graph Representation Learning. *Neural Netw.* **2024**, *173*, 106207, doi:10.1016/j.neunet.2024.106207.
8. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24, doi:10.1109/TNNLS.2020.2978386.
9. Begnardi, L.; Baier, H.; Jaarsveld, W. van; Zhang, Y. Deep Reinforcement Learning for Two-Sided Online Bipartite Matching in Collaborative Order Picking. In Proceedings of the Proceedings of the 15th Asian Conference on Machine Learning; PMLR, February 27 2024; pp. 121–136.
10. Cho, S.-H.; Shin, W.-J.; Ahn, J.; Joo, S.; Kim, H.-J. Dynamic Crane Scheduling with Reinforcement Learning for a Steel Coil Warehouse. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA); May 2024; pp. 16545–16552.
11. Xiao, Z.; Li, P.; Liu, C.; Gao, H.; Wang, X. MACNS: A Generic Graph Neural Network Integrated Deep Reinforcement Learning Based Multi-Agent Collaborative Navigation System for Dynamic Trajectory Planning. *Inf. Fusion* **2024**, *105*, 102250, doi:10.1016/j.inffus.2024.102250.
12. Ansel, J.; Yang, E.; He, H.; Gimelshein, N.; Jain, A.; Voznesensky, M.; Bao, B.; Bell, P.; Berard, D.; Burovski, E.; et al. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. *29th ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst. Vol. 2 ASPLOS 24* 2024.
13. Fey, M.; Sunil, J.; Nitta, A.; Puri, R.; Shah, M.; Stojanovič, B.; Bendias, R.; Barghi, A.; Kocijan, V.; Zhang, Z.; et al. PyG 2.0: Scalable Learning on Real World Graphs 2025.
14. Fey, M.; Lenssen, J.E. Fast Graph Representation Learning with PyTorch Geometric 2019.
15. Networkx/Networkx 2025.
16. Bellman, R. *Dynamic Programming*; Princeton University Press, 1959;
17. Christofides, N. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. **1976**.
18. Croes, G.A. A Method for Solving Traveling-Salesman Problems. *Oper. Res.* **1958**, *6*, 791–812, doi:10.1287/opre.6.6.791.
19. Lin, S.; Kernighan, B.W. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Oper. Res.* **1973**, *21*, 498–516, doi:10.1287/opre.21.2.498.
20. Goulart, F. Fillipe-Gsm/Python-Tsp 2025.
21. Čelik, N.; Škraba, A. Graph-Based Modeling of Warehouse Layouts Based on Data from Relation Database for Optimizing Order Picking Path.; Slovenian Society Informatika, Section for Operational Research, 2025; pp. 379–382.
22. Dijkstra, E.W. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* **1959**, *1*, 269–271, doi:10.1007/BF01386390.
23. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms 2017.
24. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the Proceedings of The 33rd International Conference on Machine Learning; PMLR, June 11 2016; pp. 1928–1937.
25. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation 2018.
26. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization 2019.
27. Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; Sun, Y. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification 2021.
28. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83, doi:10.2307/3001968.
29. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction 2020.

 doi:10.20944/preprints202602.0896.v1

30. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*; University of California Press, 1967; Vol. 5.1, pp. 281–298.