

Article

SL-LDA: LDA-Based Storage Location Assignment for Automated Warehouses Under MAPD Constraints [†]

Tatsuto Ito ^{1,*}, Taisei Hirayama ¹, Naoki Hattori ², Hiroki Sakaji ¹ and Itsuki Noda ¹

¹ Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0814, Japan; hirayama.taisei.p4@elms.hokudai.ac.jp (T.H.); sakaji@ist.hokudai.ac.jp (H.S.); i.noda@ist.hokudai.ac.jp (I.N.)

² Toyota Industries Corporation, Kariya 474-8601, Japan; naoki.hattori@mail.toyota-shokki.co.jp

* Correspondence: ito.tatsuto.m1@elms.hokudai.ac.jp

[†] This paper is an extended version of our paper published in the Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2025), entitled “LDAS: Proposal for Semi-optimal Storage Locations Using LDA and SA”.

Abstract

Storage location assignment in automated warehouses strongly affects order-processing efficiency. Existing co-occurrence-based approaches often rely on pointwise mutual information (PMI) statistics or direct frequency co-occurrence. This paper compares two deliberately chosen representation families for storage assignment in automated warehouses operated under Multi-Agent Pickup and Delivery (MAPD) constraints: Pointwise Positive PMI (PPPMI), representing direct pairwise co-occurrence, and Latent Dirichlet Allocation (LDA), representing latent-topic smoothing. The purpose is not to benchmark every possible representation space, but to make the pairwise-versus-latent contrast interpretable under a fixed execution pipeline consisting of task construction, visit-order selection, path planning, and collision avoidance. The broader research setting is motivated by real warehouse order data in which SKU co-occurrence structure is present, but such logs mix latent-topic effects, explicit family-based co-occurrence, noise, and demand variation. We therefore use two controlled abstractions of order structure: one generator with latent-topic mixtures and one generator with more direct family co-occurrence. We embed the proposed LDA representation and the PPPMI baseline in constrained-clustering and simulated-annealing placement methods and evaluate them against frequency-based, load-balancing, and random baselines. Evaluation is conducted in a fixed extended MAPD simulator that explicitly models orientation-aware motion, turning costs, service times, dynamic task release, and collision avoidance. In the latent-topic regime, LDA-based methods tended to form the leading group in average finite-horizon makespan, computed over completed combinations of random seeds and operating conditions. In the supplementary direct-co-occurrence condition, PPPMI was competitive in the plain representation comparison, while LDA-driven local search on top of a frequency-based initial layout remained strong. These results do not imply that LDA is universally superior; rather, they indicate that the relative suitability of PPPMI and LDA depends on the order structure and on how the representation interacts with the placement optimizer. The controlled generators are useful for isolating those effects, but they do not replace external validation on real warehouse logs.



Academic Editor: Massimo Cossentino

Received: 20 March 2026

Revised: 28 April 2026

Accepted: 14 May 2026

Published: 19 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: automated warehouse; storage location assignment; multi-agent pickup and delivery; latent Dirichlet allocation; co-occurrence modeling; simulated annealing

1. Introduction

Fulfillment centers are operational nodes within logistics and supply networks, and their efficiency directly influences end-to-end supply-chain performance. Among warehouse processes, order picking is especially critical because it aggregates the items required for each customer order and accounts for a large share of processing time and operating cost [1–3]. The growth of e-commerce has increased both order volume and demand variability, which in turn has strengthened the need for highly automated warehouse operations capable of handling frequent and short-lead-time outbound requests. Recent application-oriented studies of robotic picking and robotic warehousing similarly emphasize that algorithm design must be evaluated together with the operational environment in which humans, robots, and inventory decisions interact [4,5].

Against this background, goods-to-person automated warehouses such as Robotic Mobile Fulfillment Systems (RMFSs) have attracted considerable attention [3,6]. In RMFS-like systems, robots transport shelves or pods to picking stations, where workers or automated equipment retrieve the requested items. This architecture reduces worker travel and improves responsiveness, but it also creates persistent multi-agent interference because many robots move continuously in a shared space. Path conflicts, aisle congestion, and waiting times can therefore dominate operational performance. Such settings are naturally related to Multi-Agent Pickup and Delivery (MAPD), where multiple agents repeatedly serve transport requests that arrive over time [7–9].

However, warehouse performance is not determined only by online task allocation and collision-free path planning. Storage location assignment, namely, the decision of which storage slot holds each SKU, also has a large effect on robot travel distance, congestion, service time, and finite-horizon completion time. Placing high-frequency SKUs in accessible positions or locating SKUs that are frequently requested together near one another can directly improve outbound efficiency. Accordingly, storage location assignment has long been studied as the Storage Location Assignment Problem (SLAP) [1,10–12].

Prior SLAP studies have considered frequency-based, class-based, distance-based, and load-balancing policies, as well as approaches that exploit co-occurrence among products requested in the same order [13,14]. Co-occurrence-based placement is attractive because it can reflect order structure that is difficult to capture through single-SKU frequency alone. At the same time, the co-occurrence representations used so far have mostly been based on pointwise mutual information (PMI) variants, direct pair counts, or rule-based associations. The use of latent topic representations, which are standard in text mining, remains underexplored for warehouse storage assignment.

This paper therefore studies Latent Dirichlet Allocation (LDA) as a co-occurrence representation for storage assignment in automated warehouses and compares it with Pointwise Positive PMI (PPPMI). Although LDA was originally proposed as a probabilistic model for extracting latent topic structure from document collections [15,16], it can also be applied to order data by treating each order as a “document” and each SKU as a “word”. Under this interpretation, LDA may capture smoothed latent relations among SKUs that differ from the more literal pairwise associations emphasized by PMI-based features.

The comparison is intentionally narrow. Our purpose is not to benchmark every available representation family, but to contrast two interpretable modeling choices under the same execution pipeline, by which we mean the fixed sequence of task construction, visit-order selection, motion planning, reservation handling, and collision avoidance used to execute a placement: PPPMI as a representation of directly observed pairwise co-occurrence, and LDA as a representation that smooths co-occurrence through latent topics. Alternatives such as nonnegative matrix factorization (NMF) [17], correlated topic models (CTM) [18],

and BERT-based embeddings [19] are also plausible candidates, but they are outside the present scope and remain for future work.

The broader research setting is motivated by practical warehouse order data, in which SKU co-occurrence structure is present. However, real logs can mix latent topic-like regularities, explicit family-based co-occurrence, noise, and demand variation. If all of these effects are left entangled, a PPPMI-versus-LDA difference is difficult to interpret. For that reason, this journal version uses two controlled order generators: a latent-mixture generator (*topic*) and a direct family-co-occurrence generator (*basket*). These generators should be read as controlled abstractions of co-occurrence patterns suggested by real orders, not as a substitute for eventual validation on real warehouse logs.

Our goal is not to claim that LDA is always better than PPPMI. Rather, we ask when a latent-topic representation is better matched to the order structure and whether that representational fit propagates to execution performance when the only experimental factor varied is the storage assignment policy. To keep the comparison interpretable, the main experiment uses orders with explicit latent topic structure, and a supplementary direct-co-occurrence condition (*basket*) checks how the conclusions change when co-occurrence is dominated more directly by explicit pairwise association. Concretely, we compare PPPMI and LDA as co-occurrence representations, embed them in both clustering-style and local-search-style placement methods, and compare them with classical baselines that do not use co-occurrence information. Evaluation is performed in an extended MAPD simulator with a fixed planner that includes orientation-aware motion, turning costs, loading/unloading service times, and collision avoidance. This isolates the impact of storage assignment under a realistic execution pipeline rather than under a purely geometric distance surrogate.

The main questions addressed in this paper are as follows. First, under order streams with latent-topic structure, is LDA more effective than PPPMI as a co-occurrence representation for storage assignment? Second, is the effect of the representation consistent across clustering-style and local-search-style placement methods, or does it depend on the interaction with the initial solution, the operating conditions, and the order structure itself?

The contributions of this paper are threefold:

1. **Methodological contribution:** we evaluate storage assignment in the context of extended MAPD with a fixed planner that explicitly models orientation-aware motion, turning costs, service times, and collision avoidance, thereby providing a framework for comparing placement policies through execution performance in automated warehouses.
2. **Positioning and comparison contribution:** we frame the study as an interpretable comparison between PPPMI as a direct pairwise representation and LDA as a latent smoothing representation, and complement the main topic-structured condition with a *basket* ablation so that the representation effect can be interpreted in terms of order structure rather than treated as a generic benchmark ranking.
3. **Empirical contribution:** in the controlled topic-structured regime, LDA-based methods tended to form the leading group in mean makespan, whereas in the direct-co-occurrence condition (*basket*) PPPMI became competitive in the plain representation comparison; thus the results point to structure-dependent suitability of PPPMI versus LDA rather than to universal dominance of either representation.

Note on Prior Publication

Parts of this study build on our PAAMS 2025 conference paper [20]. The conference version reported an earlier storage-assignment study under a simpler execution setting, where immediate task assignment was handled by a greedy strategy and the empirical scope was more limited. The present journal manuscript substantially extends that work by replacing the earlier execution setting with a more realistic warehouse-oriented execution

pipeline, referred to here as SHARP [21], built on the Anytime Multi-Task MAPD simulator framework [22]. Here, the execution pipeline denotes the complete fixed implementation used during evaluation, including task construction, visit-order generation, orientation-aware path search, reservation-table collision avoidance, and service-time handling. It also broadens the evaluation to multiple layouts, load conditions, goal policies, noise levels, and a larger comparison set, and it adds the PPPMI-versus-LDA analysis, the initial-solution ablations, and the supplementary order-structure experiments.

2. Related Work

This section reviews prior work on MAPD, SLAP, and co-occurrence modeling from the viewpoint most relevant to this paper: how to evaluate storage assignment while accounting for multi-agent interference during execution.

2.1. Multi-Agent Pickup and Delivery

MAPD studies how multiple agents should repeatedly serve transport tasks while maintaining collision-free motion. Representative methods include Token Passing and related schemes, and more recent work has improved scalability and congestion handling in large systems [6–9,23–25]. Most of this literature assumes that pickup locations are already given and focuses on online assignment and planning rather than on how storage locations should be designed. An exception more closely related to our setting is the work of Hattori et al. [14], which reported MAPD execution improvements through capacity-constrained clustering with co-occurrence information. Our study follows this direction by keeping the planner fixed and comparing storage assignment designs instead.

2.2. Storage Location Assignment Problem

The SLAP literature covers classical policies such as random storage, frequency-based storage, and class-based storage, as well as metaheuristic optimization for assignment refinement [1,2,11,26]. Closest to this paper are correlated-storage approaches that place related SKUs near each other by exploiting order-level associations [12,13,27,28]. Recent practical robotic-warehousing studies also show the importance of evaluating inventory, picking, and robot-execution decisions in an integrated operational setting rather than as isolated geometric subproblems [4,5]. However, many SLAP studies still evaluate policies mainly through travel distance for one or a few pickers. A comparatively limited number of studies incorporate MAPD-specific interference, collision avoidance, and shared-space congestion directly into the evaluation loop.

2.3. Co-Occurrence Analysis for Order Data

Pointwise mutual information (PMI), positive PMI (PPMI), and frequent-itemset mining are widely used to represent correlations in transactional data [29–31]. A PMI-derived representation describes each SKU by how strongly it co-occurs with the other SKUs in the order history. In this paper, PPPMI denotes the nonnegative SKU-feature matrix obtained from positive PMI with smoothing, so it functions as a direct feature representation of observed pairwise co-occurrence. Hattori et al. [14] used PMI-based similarity features in combination with capacity-constrained clustering and demonstrated placement improvements in automated warehouse settings. At the same time, pairwise methods rely heavily on directly observed pairs and therefore leave open the question of how to smooth sparse or unobserved co-occurrences.

2.4. Latent Dirichlet Allocation

LDA [15,16] is a generative model that represents discrete observations as mixtures of latent topics. It has also been applied to short transactional data such as purchase histories

and market baskets [32–34]. The appeal of LDA in our setting is that it can represent not only direct pairwise co-occurrence but also higher-order latent structure shared across several SKUs. Nevertheless, its robustness on short orders and, more importantly, its contribution to actual storage-assignment performance in automated warehouses remain insufficiently understood.

2.5. Research Gap and Positioning

The literature therefore leaves three related gaps. First, MAPD research and SLAP research are still largely separate, and only a limited number of studies evaluate the interaction between placement design and multi-agent interference in a unified experimental setting that resembles practical robotized warehouses. Second, PPPMI has been tested as a co-occurrence representation for storage assignment, but the conditions under which latent representations such as LDA are advantageous are not yet clear. Answering that question requires controlling the order structure; otherwise an observed PPPMI-versus-LDA difference is hard to interpret. Third, the goal of the present paper is not an exhaustive benchmark across every modern representation family. Alternatives such as NMF [17], CTM [18], and BERT-based embeddings [19] are also plausible candidates, but including all of them would blur the more interpretable question addressed here: how a direct pairwise representation compares with a latent smoothing representation under the same execution pipeline. This paper therefore addresses the first two gaps by positioning itself as a deliberately controlled comparison between PPPMI and LDA within a shared extended MAPD simulator while disentangling the roles of the representation, the initial solution, local search, and order-generation structure. Broader representation benchmarking is left for future work.

3. Problem Setting

3.1. Evaluation Setting as an Extended MAPD Problem

We treat automated warehouse operation with storage assignment as an extended MAPD problem. Let the warehouse environment be

$$\mathcal{E}_{\text{wh}} = \langle G, A \rangle,$$

where $G = \langle V, E_G \rangle$ is the movement graph induced by the warehouse grid and $A = \{a_1, \dots, a_m\}$ is the set of agents. Let $\mathcal{O}$ denote the order stream, $\mathcal{I}$ the set of SKUs, and $\mathcal{S}$ the set of storage slots. The storage assignment is represented by

$$p : \mathcal{I} \rightarrow \mathcal{S}.$$

Figure 1 illustrates the warehouse environment used by the simulator on the well-formed layout. The traversable cells form

$$V = V_O \cup V_P \cup V_U \cup V_L \cup V_S,$$

and edges in E_G connect adjacent traversable cells. Because each $v \in V_P$ has multiple shelf levels h , an assignment $p(i) = (v, h)$ specifies both the access cell and the shelf level behind that cell.

well-formed layout

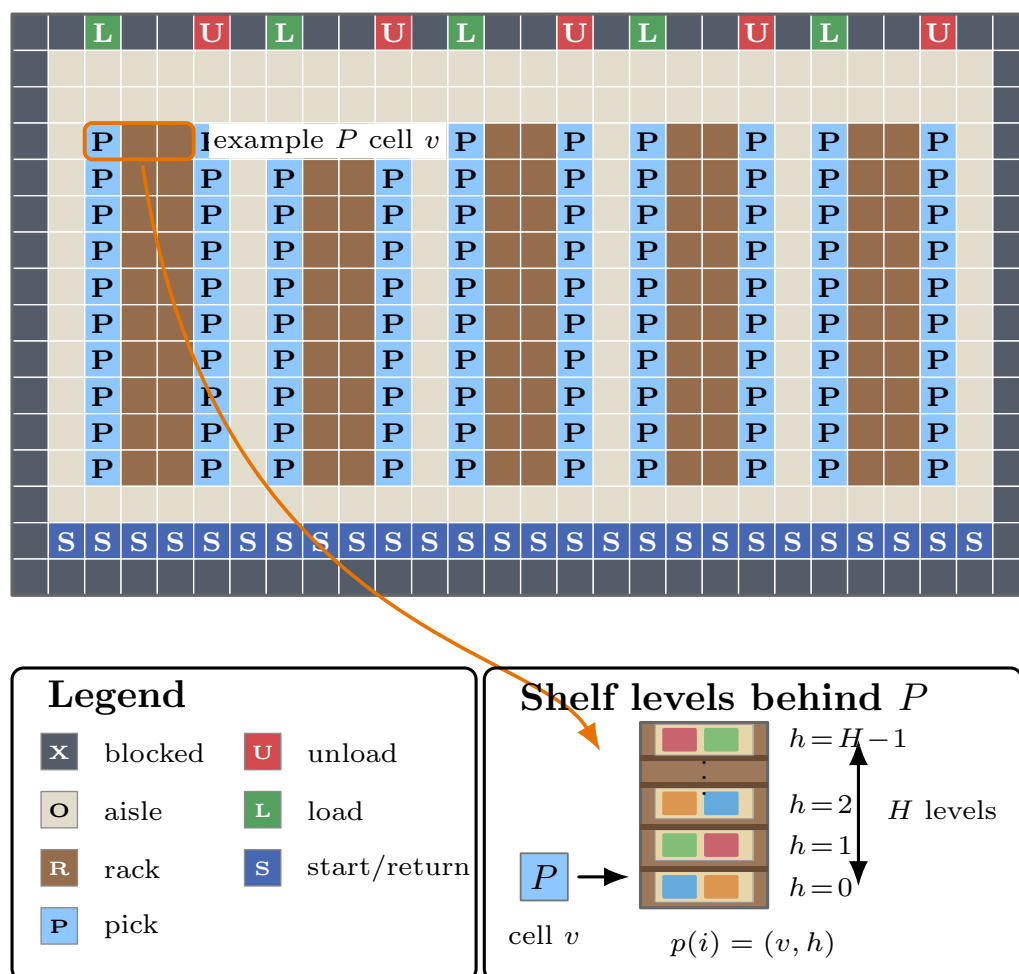


Figure 1. Schematic warehouse environment based on the simulator’s well-formed layout. The movement graph $G = \langle V, E_G \rangle$ is constructed from traversable cells, and the storage slots are defined as $S = \{(v, h) \mid v \in V_p, h \in \{0, \dots, H-1\}\}$, namely, a pair of a pick-access cell and a shelf level. The lower-left panel gives the legend for the cell symbols **X**, **O**, **R**, **P**, **U**, **L**, and **S**, and the lower-right panel illustrates how shelf levels h are attached behind an example **P** cell v .

The joint optimization problem of agent execution and storage assignment can then be written as

$$\langle \pi_{\mathcal{A}}^*, p^* \rangle = \arg \min_{\pi_{\mathcal{A}}, p} \text{MS}(\pi_{\mathcal{A}}(\Gamma(\mathcal{O}, p), \mathcal{E}_{\text{wh}})), \quad (1)$$

where $\Gamma(\mathcal{O}, p)$ maps the order stream and a storage assignment to an MAPD task sequence and $\text{MS}(\cdot)$ denotes makespan. In this paper, makespan means the completion time of the last order request released within the finite evaluation horizon, that is, the elapsed time until all released orders have been processed. Jointly optimizing online MAPD decisions and storage assignment is computationally difficult, so in this paper we fix the agent policy $\pi_{\mathcal{A}}$ and compare only the storage assignment:

$$p^* = \arg \min_p \text{MS}(\pi_{\mathcal{A}}^{\text{fixed}}(\Gamma(\mathcal{O}, p), \mathcal{E}_{\text{wh}})). \quad (2)$$

Equations (1) and (2) distinguish the full joint problem from the fixed-planner storage-assignment subproblem addressed here. This lets us compare placement methods through the execution performance they induce under a realistic, fixed planner. Because real

warehouses operate continuously, makespan is not used here as a standalone long-run operational KPI. It is a finite-horizon dynamic-simulation metric: orders are released during the evaluation horizon, the simulator continues until all released order requests are completed or the maximum simulation length is reached, and the reported makespan is the last completion time. The mean makespan reported later is the arithmetic mean of these run-level makespans over the completed combinations of random seeds and operating conditions.

3.2. Warehouse Layout and Storage Slots

The warehouse is represented as a rectangular grid. Table 1 summarizes the meaning of each cell type. Traversable cells are divided into V_O , V_P , V_U , V_L , and V_S , and the set of outbound goal cells is denoted by $\mathcal{U} = V_U$. SKU items are stored in racks behind P cells, and each P location $v \in V_P$ has H shelf levels. The set of storage slots is therefore

$$\mathcal{S} = \{(v, h) \mid v \in V_P, h \in \{0, \dots, H-1\}\}.$$

The level index h distinguishes storage positions vertically, while robot motion is still executed on the two-dimensional grid.

Table 1. Meaning of the grid-cell symbols.

Symbol	Meaning
S	Dedicated initial position and return location of each agent
P	Rack-front access cell used to pick an item
U	Unloading position for outbound service
L	Loading position for inbound service
O	Traversable aisle cell
R	Rack cell
X	Blocked cell

The simulator contains at least two layout types:

- **Non-well-formed layout:** shared aisles are narrow and passing opportunities are limited, so congestion is likely to cause head-on blocking.
- **Well-formed layout:** shared aisles are wide and intersections permit easier avoidance behavior, making it possible to test whether a placement method remains effective under more permissive geometry.

Our use of the term “well-formed” follows the well-formed infrastructure condition of Čap et al. [35]: any pair of endpoints is connected by a path that avoids other endpoints sufficiently so that robots resting at endpoints do not completely block the system. The non-well-formed layout does not satisfy this condition and is instantiated here as a narrow-aisle setting with concentrated interference. By using both layout types, we can test whether the placement effects generalize beyond one specific warehouse geometry.

3.3. Order Generation and Order-Request Construction

Orders are generated at discrete timesteps $t = 1, \dots, T_{\text{gen}}$. At each timestep, an order is generated with probability ρ . Order size is drawn from

$$\text{Unif}\{m_{\min}, \dots, m_{\max}\},$$

and the main experiments use $m_{\min} = 1$ and $m_{\max} = 10$.

In the experimental setting used here, the SKU catalog contains $N = 600$ items partitioned into multiple product families, meaning groups of related SKUs that tend to appear in similar order contexts. Within each family, *core* items act as the central representative

SKUs of the group, *halo* items are neighboring SKUs that are somewhat less central but still likely to be chosen together with the core items, and weak bridge relations connect neighboring families. This family structure is shared by the latent-structure generator used in the main experiments (topic) and by the direct-co-occurrence generator used in Appendix A.4 (basket).

The order generators are used because the purpose of this paper is to compare co-occurrence representations under controlled structure rather than on a single mixed dataset. The broader research setting is informed by real warehouse order data, where SKU co-occurrence structure is present, but such logs can combine latent mixture structure, explicit family-based co-occurrence, noise, and demand variation. If training and evaluation orders mix several mechanisms simultaneously, a PPPMI-versus-LDA difference is difficult to interpret. The main experiment therefore uses the latent-structure generator (topic) as a controlled regime for testing a latent representation, while Appendix A.4 uses the direct-co-occurrence generator (basket) as a counterpoint in which explicit family-level association is stronger and PMI-based statistics may be better matched. The generators are thus intended as controlled abstractions of co-occurrence structure suggested by real orders, not as toy settings and not as replacements for final validation on real warehouse logs. In particular, they let us vary whether the order stream is driven more by latent topic mixtures or by more explicit within-group co-occurrence, together with order size, arrival rate, and co-occurrence degradation through noise, while keeping the SKU catalog, planner, and warehouse layouts fixed.

Under the latent-structure generator (topic), each order is sampled from a topic–SKU distribution Φ that is consistent with the family structure:

$$\theta_t \sim \text{Dir}(\alpha_{\text{topic}} \mathbf{1}), \quad z_{tn} \sim \text{Cat}(\theta_t), \quad i_{tn} \sim \text{Cat}(\Phi_{z_{tn}}).$$

Here, θ_t is the order-level topic mixture for order t , z_{tn} is the topic assigned to the n -th SKU token in that order, i_{tn} is the sampled SKU, and Φ is the topic–SKU distribution. In the main experiment, we fix $\alpha_{\text{topic}} = 0.3$. Hence each order is generated as a mixture of latent topics. This provides a controlled condition under which we can test whether a latent representation yields an execution advantage when the order stream truly contains latent mixture structure. The direct-co-occurrence generator (basket) is treated as an order-data ablation in Appendix A.4. Both generators include a noise rate ε , which replaces a fraction of sampled SKUs with other SKUs in order to control co-occurrence degradation.

Training and evaluation order streams are generated independently and are then converted into simulator order requests according to the storage assignment p . Each order request is simply the simulator representation of one outbound order: it specifies when the order appears, which pick-access cells must be visited, and which unloading destination cell must be reached. For an order o_t , the corresponding order request is written as

$$\tau_t = (r_t, \mathcal{C}_t, u_t),$$

where r_t is the release timestep, $\mathcal{C}_t$ is the set of rack-front access cells corresponding to the ordered SKUs under assignment p , and u_t is the outbound goal. The word “goal” is used in the MAPD sense of a destination cell for unloading, not as a performance objective. We use two goal policies. Under random, u_t is sampled uniformly from the outbound goals. Under nearest, the unloading cell minimizing the total travel distance from the ordered SKU access cells is selected. Consistent with the shared evaluation settings, the reported experiments use fixed service times of 5 for picking, unloading, and loading; in particular, the picking dwell time is 5 for every SKU.

3.4. Agent Motion Model and Operational Costs

The state of each agent a is

$$x_a(t) = (v_a(t), d_a(t)), \quad d_a(t) \in \{N, E, S, W\},$$

that is, the simulator tracks both position and orientation. This is necessary because rack access direction and turning time are modeled explicitly.

The primitive actions are forward motion, $\pm 90^\circ$ rotation, and waiting. Their unit costs are

$$c_{\text{move}} = 1, \quad c_{\text{wait}} = 1, \quad c_{\text{rot}} = \tau_{\text{rot}}.$$

The default value is $\tau_{\text{rot}} = 3$, so turning is more expensive than one straight step. Hence the evaluation depends on orientation-aware travel cost rather than only on Manhattan distance.

Service actions also incur explicit dwell times. Picking SKU c requires

$$\tau_{\text{pick}}(c) = 5,$$

unloading at an outbound location requires

$$\tau_{\text{unload}} = 5,$$

and loading at an inbound location requires

$$\tau_{\text{load}} = 5.$$

Thus, all service times used in the reported experiments are fixed at 5. Modeling turning, picking, unloading, and loading separately prevents the placement evaluation from collapsing to a pure shortest-distance objective.

The MAPD execution constraints are:

1. No two agents may occupy the same cell at the same time (no vertex conflict).
2. No two agents may swap cells across consecutive timesteps (no edge-swap conflict).
3. A home S cell assigned to another agent is not traversable and remains exclusive to its owner.
4. Cells P, U, and L remain reserved during service dwell times.

3.5. Fixed Planner: Base Simulator and SHARP Execution Layer

The evaluation pipeline in this paper uses the simulator setting of Anytime Multi-Task MAPD under energy constraint [22] as its base framework. On top of that base framework, we use a warehouse-oriented execution layer, referred to here as SHARP [21]. In other words, Anytime Multi-Task MAPD is the underlying simulator setting, whereas SHARP is the stronger execution model used in this study for placement evaluation. The purpose of the present work is not to improve task allocation or path planning themselves, but to compare storage assignment under a common execution pipeline. For this reason, the simulator uses the same deterministic task-release stream and the same fixed execution rules for every placement method. When an agent becomes available, the fixed task-assignment rule selects the next released order request according to the base simulator configuration; the visit-order module then enumerates candidate orders for visiting the pick-access cells in that request, and the planner executes the selected order while respecting the reservation table. Relative to the simpler conference setting that used greedy immediate-task assignment, SHARP retains the same base simulator configuration while adding the warehouse-specific execution rules needed in this study, such as orientation-aware motion costs, service times at P/U/L cells, and route execution that can insert waiting or detours

through reservation-aware planning. All of these assignment and planning choices are fixed across methods; no storage-assignment method is allowed to change them.

The planning layer used in the experiments consists of candidate visit-order generation within a task, orientation-aware static A* cost computation [36], and collision avoidance through Safe Interval Path Planning (SIPP) with a reservation table [37]. In single-agent planning, both position and orientation are part of the state; forward cost is 1 and turn cost is τ_{rot} . When the agent services P, U, or L, the corresponding cell remains occupied for the full service duration. SIPP constructs safe intervals from the reservation table and inserts only the waiting needed to avoid vertex and edge-swap conflicts. This gives a realistic but fixed planner with which to compare storage assignments while keeping the underlying simulator consistent across methods.

For reproducibility, the simulation loop is as follows. At each timestep, newly released orders are converted to order requests according to the current storage assignment. When an agent is available, the fixed assignment rule selects one released order request; the visit-order module enumerates candidate orders of the required pick-access cells; the orientation-aware A* module computes static route costs for those candidates; and SIPP generates a reservation-table-consistent route for the selected sequence. The agent then executes forward moves, rotations, waits, and service dwell actions, while occupied cells are recorded in the reservation table. A run terminates when all order requests released within the finite evaluation horizon have been completed or when the maximum simulation length is reached. This loop is identical for all storage-assignment methods, so the reported differences arise from the placement p rather than from changes in task allocation or path planning.

3.6. Fixed Elements in the Comparison Design

The only experimental factor varied in this paper is the storage assignment method. The following components are fixed across all compared conditions:

1. the order-generation rules and random-seed handling;
2. the goal policy (random or nearest);
3. the visit-order enumerator within each order request;
4. the orientation-aware A* and reservation-table SIPP planner;
5. the turning cost and all service-time parameters.

Consequently, performance differences in the experiments indicate how strongly changes in SKU placement propagate to MAPD execution under realistic costs and collision avoidance.

4. Method

This section formulates the storage assignment methods. Let the SKU set be $\mathcal{I} = \{1, \dots, N\}$ and the training-order set be $\mathcal{D} = \{o_1, \dots, o_M\}$. Each order o_d is treated as a set of SKUs, and $\mathbf{1}[i \in o_d]$ indicates whether SKU i appears in order d . Let $\mathcal{S}$ be the set of storage slots and $\mathcal{U}$ the set of outbound goals. Each slot $s \in \mathcal{S}$ has position (x_s, y_s) , shelf level h_s , and a fixed goal-zone assignment

$$g(s) = \bar{g}(s), \quad s \in \mathcal{S},$$

where $\bar{g} : \mathcal{S} \rightarrow \mathcal{U}$ is randomly generated before the experiments and then shared across all compared methods. We write $\mathcal{S}_u = \{s \in \mathcal{S} \mid g(s) = u\}$ for the zone corresponding to goal u and $C_u = |\mathcal{S}_u|$ for its capacity. The storage assignment $p : \mathcal{I} \rightarrow \mathcal{S}$ is a bijection.

We compare the methods ck-pmi, ck-lda, sa-pmi, sa-lda, pmi-frequency-nearest, and lda-frequency-nearest alongside three classical baselines. Table 2 summarizes the compared methods and the intended comparison axes.

Table 2. Compared methods and their roles in the comparison.

Method	Comparison Purpose
random	Baseline without co-occurrence information
frequency-nearest	Classical baseline that places frequent SKUs close to goals
balance	Classical baseline that uses load balancing only
ck-pmi	PPPMI with CK-style capacity-constrained clustering
ck-lda	LDA with CK-style capacity-constrained clustering
sa-pmi	Added effect of simulated annealing after ck-pmi
sa-lda	Added effect of simulated annealing after ck-lda
pmi-frequency-nearest	Interaction between a frequency-based initial solution and PMI-driven simulated annealing
lda-frequency-nearest	Interaction between a frequency-based initial solution and LDA-driven simulated annealing

4.1. Proxy Objective

The ultimate objective is to improve makespan under the fixed planner. Directly searching over storage assignments using simulated makespan as the objective is computationally expensive, because each candidate assignment p requires a full dynamic MAPD simulation with task releases, service times, and reservation-table collision avoidance. We therefore introduce a surrogate that encourages SKUs that co-occur frequently to be placed near each other. The surrogate is based on the following operational rationale rather than on a formal optimality proof. If SKUs that tend to appear in the same order are assigned to nearby access cells, then the set of pick cells visited within one order becomes less spatially dispersed. Under the fixed execution policy, lower dispersion tends to shorten between-pick travel, reduce the time for which agents occupy productive aisle segments near those picks, and thereby reduce reservation conflicts and waiting cascades. The resulting placement is therefore expected to reduce both per-order service time and congestion-induced delay, which moves makespan in the favorable direction.

For slots $s, t \in \mathcal{S}$, define

$$d(s, t) = |x_s - x_t| + |y_s - y_t| + \lambda_h |h_s - h_t|, \quad (3)$$

where $\lambda_h \geq 0$ controls the contribution of shelf-height difference. Under assignment $p : \mathcal{I} \rightarrow \mathcal{S}$, the slot distance between SKUs i and j is

$$d_{ij}(p) := d(p(i), p(j)).$$

Let $w_{ij} \geq 0$ be a co-occurrence weight representing how strongly SKUs i and j tend to be requested together. We then consider the weighted distance sum

$$L(p) = \sum_{i < j} w_{ij} d_{ij}(p). \quad (4)$$

This quantity measures the average spatial dispersion of strongly co-occurring SKU pairs and serves as a proxy for how spread out the visited pick cells of one order will be. Equations (3) and (4) define the distance and weighted-dispersion terms used by this surrogate. Reducing it should shorten intra-order movement and narrow the temporal occupancy window created by one task.

If the per-order processing time $T(o; p)$ can be approximated by a sum of fixed processing cost, within-order movement, and congestion delay proportional to that movement, then

$$\mathbb{E}_{o \sim \mathcal{D}}[T(o; p)] \lesssim \text{const} + (\beta + \gamma)L(p). \quad (5)$$

Here, β summarizes the sensitivity of within-order travel time to the weighted dispersion $L(p)$, while γ summarizes the sensitivity of congestion delay to the same travel-related occupancy. Equation (5) is not claimed as a rigorous bound because actual makespan also depends on task interleaving, route conflicts, and planner decisions. Rather, it formalizes the directional argument above: concentrating within-order SKU locations reduces travel and aisle occupancy, and lower occupancy tends to reduce MAPD contention and hence makespan. Section 6.5 later checks this claim empirically by comparing the surrogate with measured mean makespan. The methods below therefore derive SKU embeddings for clustering and pair weights for simulated annealing from PMI- and LDA-based co-occurrence representations.

4.2. Co-Occurrence Representations (PMI/LDA)

Define SKU order frequency and pairwise co-occurrence count by

$$f_i = \sum_{d=1}^M \mathbf{1}[i \in o_d], \quad c_{ij} = \sum_{d=1}^M \mathbf{1}[i \in o_d] \mathbf{1}[j \in o_d].$$

PMI-based statistics are computed from set-based frequencies, that is, repeated appearances of the same SKU within one order are collapsed.

4.2.1. PPPMI Representation

For PMI-based methods, we first compute

$$p(i) = \frac{f_i}{M}, \quad p(i, j) = \frac{c_{ij}}{M}.$$

We smooth the context-side marginal as

$$q(j) = \frac{f_j^{\alpha_{\text{ctx}}}}{\sum_{\ell=1}^N f_{\ell}^{\alpha_{\text{ctx}}}}, \quad \alpha_{\text{ctx}} = 0.75,$$

which mitigates the tendency of raw PMI to overweight low-frequency events and follows the context-distribution smoothing practice shown to be effective in distributional similarity and PPPMI-based representations [38]. The asymmetric PMI is

$$\text{PMI}_{i \rightarrow j} = \log \frac{p(i, j)}{p(i) q(j)},$$

and the symmetric positive PMI matrix $M^{\text{PMI}} \in \mathbb{R}_{\geq 0}^{N \times N}$ is defined by

$$M_{ij}^{\text{PMI}} = \max\left(0, \frac{\text{PMI}_{i \rightarrow j} + \text{PMI}_{j \rightarrow i}}{2}\right), \quad M_{ii}^{\text{PMI}} = 0. \quad (6)$$

Equation (6) defines the PPPMI value used as the direct pairwise co-occurrence representation. The PMI embedding of SKU i is the row vector

$$e_i^{\text{PMI}} = M_{i:}^{\text{PMI}}.$$

4.2.2. LDA Representation

For LDA-based methods, each order is treated as a document and each SKU as a word. We train LDA with K topics, Dirichlet priors

$$\alpha_{\text{LDA}} = 0.3, \quad \beta_{\text{LDA}} = 0.05,$$

and collapsed Gibbs sampling. These hyperparameters are fixed across all conditions; sensitivity analysis for them is beyond the scope of this paper. The value $\alpha_{\text{LDA}} = 0.3$ encourages each order to concentrate on a small number of topics, while $\beta_{\text{LDA}} = 0.05$ keeps topic-SKU distributions sparse enough to preserve family-level structure without assigning zero probability to rare SKUs. Thus, the LDA parameters are used to obtain a stable latent representation for the placement comparison rather than to tune the simulator outcome. For the n th token in document d taking SKU i , the topic-assignment update is

$$P(z_{dn} = k \mid z_{-dn}, w_{dn} = i, w_{-dn}) \propto \left(n_{dk}^{-dn} + \alpha_{\text{LDA}} \right) \frac{n_{ki}^{-dn} + \beta_{\text{LDA}}}{n_k^{-dn} + N\beta_{\text{LDA}}}, \quad (7)$$

where n_{dk} is the number of times topic k is assigned in document d , n_{ki} is the number of times SKU i is assigned to topic k , and $n_k = \sum_i n_{ki}$.

After training, the topic-word distribution and topic prior are estimated as

$$\phi_{ki} = \frac{n_{ki} + \beta_{\text{LDA}}}{n_k + N\beta_{\text{LDA}}}, \quad \pi_k = \frac{n_k + \alpha_{\text{LDA}}}{\sum_{\ell=1}^K (n_{\ell} + \alpha_{\text{LDA}})}.$$

The SKU embedding used in CK-style clustering is the posterior topic distribution of SKU i ,

$$e_{ik}^{\text{LDA}} = \frac{\pi_k \phi_{ki}}{\sum_{\ell=1}^K \pi_{\ell} \phi_{\ell i}}. \quad (8)$$

For the simulated-annealing objective, we also construct the shared-topic pair matrix

$$Q_{ij}^{\text{LDA}} = \sum_{k=1}^K \pi_k \phi_{ki} \phi_{kj}, \quad Q_{ii}^{\text{LDA}} = 0. \quad (9)$$

Equations (7), (8), and (9) respectively define the Gibbs update, SKU embedding, and shared-topic pair matrix used by the LDA-based methods.

Representational Difference Between PMI and LDA

PMI directly scores observed pairwise co-occurrence using only c_{ij} and the marginals f_i, f_j . Unobserved pairs therefore receive no positive evidence. By contrast, LDA infers higher-order latent structure through a shared topic variable. Even when direct co-occurrence is sparse, two SKUs may still obtain similar embeddings or positive pair weight if they are likely under the same topic. The PMI-versus-LDA comparison in this paper can thus be interpreted as a comparison between faithful use of observed pair co-occurrence and smoothed latent-topic modeling.

4.3. Baseline Placement Methods

We first compare three classical baselines: random, frequency-nearest, and balance. These methods are easy to interpret and represent three different design principles without explicit co-occurrence modeling: random placement, placing high-frequency SKUs near goals, and balancing load across zones.

4.3.1. Random

The random method ignores co-occurrence entirely. It shuffles the SKU set and the slot set independently and matches them in order. Equivalently,

$$p_{\text{rand}} \sim \text{Unif}(\Pi(\mathcal{I}, \mathcal{S})),$$

where $\Pi(\mathcal{I}, \mathcal{S})$ is the set of all bijections from $\mathcal{I}$ to $\mathcal{S}$. This provides a neutral baseline against which the benefit of co-occurrence-aware methods can be assessed.

4.3.2. Frequency-Nearest

The frequency-nearest baseline sorts SKUs by descending order frequency and assigns them to slots sorted by increasing average distance to the outbound goals. Frequent SKUs are therefore prioritized for easy-to-access slots. This is a classical turnover-based storage policy that does not use co-occurrence information.

4.3.3. Balance

The balance baseline aims to distribute the number of SKUs approximately evenly across goal zones. After shuffling the SKUs, it assigns them to zones in round-robin order. The within-zone order remains random, so neither frequency nor co-occurrence is used.

4.4. CK-Style Placement

CK-Style Capacity-Constrained Clustering

The ck-* initial solutions are generated by a CK-style capacity-constrained clustering heuristic. The input is an SKU embedding e_i and a frequency value f_i for each SKU. First, the embeddings are normalized as

$$\tilde{e}_i = \frac{e_i}{\|e_i\|_2}.$$

Next, $|\mathcal{U}|$ seeds are chosen by

$$s_1 = \arg \max_{i \in \mathcal{I}} f_i,$$

and, for $t \geq 2$,

$$s_t = \arg \max_{i \in \mathcal{I} \setminus \{s_1, \dots, s_{t-1}\}} (f_i + 1) \left(1 - \max_{r < t} \tilde{e}_i^\top \tilde{e}_{s_r} \right). \quad (10)$$

The first factor prefers high-frequency SKUs, and the second factor discourages selecting seeds too similar to earlier ones. The $+1$ term prevents the frequency factor from becoming zero for tail SKUs, so seed selection remains jointly influenced by frequency and representational diversity even for rarely ordered items.

Each zone $u \in \mathcal{U}$ is then paired with one seed, and SKUs are processed in descending frequency order. An SKU is assigned to the feasible zone with remaining capacity that maximizes

$$a(i) = \arg \max_{u: |A_u| < C_u} \tilde{e}_i^\top \tilde{e}_{s(u)}, \quad (11)$$

breaking ties in favor of the zone with lower fill ratio $|A_u|/C_u$, where A_u is the current set of SKUs assigned to zone u and C_u is its capacity. Thus, when two zones are equally similar for a given SKU, the less-filled zone is preferred. For the set A_u assigned to zone u , we compute the centroid

$$\mu_u = \frac{1}{|A_u|} \sum_{i \in A_u} \tilde{e}_i,$$

then sort SKUs within the zone so that high-frequency SKUs similar to the centroid appear earlier. This ordering is mapped to the slots in $\mathcal{S}_u$ sorted by proximity to goal u . For example, suppose there are two zones u_1 and u_2 with capacity 2 each and three SKUs i_1, i_2, i_3 processed in descending frequency order. If i_1 and i_2 become the two seeds, then i_3 is assigned to the zone whose seed has the larger cosine similarity to $\tilde{e}_{i_3}$; if the similarities tie, the algorithm chooses the zone with the smaller current fill ratio. After zone membership is fixed, the SKUs inside each zone are ordered using frequency and centroid similarity and then mapped to the nearer slots of that zone.

ck-pmi

ck-pmi uses $e_i = e_i^{\text{PMI}}$ in Equations (10) and (11). Hence each row of the PPPMI matrix is treated as the direct co-occurrence profile of a SKU, and SKUs with similar profiles are clustered into the same zone.

ck-lda

ck-lda uses $e_i = e_i^{\text{LDA}}$ instead. It therefore groups SKUs according to similarity in posterior topic distribution rather than direct pairwise co-occurrence. The comparison between ck-pmi and ck-lda isolates the effect of the SKU representation under the same clustering mechanism.

4.5. SA-Style Placement

4.5.1. Objective Function for Simulated Annealing

For SA-style methods, the slot distance in Equation (3) is transformed into a slot closeness score

$$C_{st} = \frac{1}{1 + d(s, t)}, \quad C_{ss} = 0, \quad (12)$$

Equation (12) is then combined with the pair-weight definitions below, and we use $\lambda_h = 0.1$. To emphasize frequent SKUs without letting frequency dominate the co-occurrence signal, we normalize frequency by

$$\bar{f}_i = \frac{f_i}{\max_j f_j}.$$

The PMI-based pair weight is

$$W_{ij}^{\text{PMI}} = \left(M_{ij}^{\text{PMI}} + \max \left(0, \frac{\langle e_i^{\text{PMI}}, e_j^{\text{PMI}} \rangle}{\|e_i^{\text{PMI}}\|_2 \|e_j^{\text{PMI}}\|_2} \right) \right) \sqrt{\bar{f}_i \bar{f}_j}, \quad W_{ii}^{\text{PMI}} = 0, \quad (13)$$

and the LDA-based pair weight is

$$W_{ij}^{\text{LDA}} = Q_{ij}^{\text{LDA}} \sqrt{\bar{f}_i \bar{f}_j}, \quad W_{ii}^{\text{LDA}} = 0. \quad (14)$$

The cosine-similarity term in Equation (13) complements the direct PMI entry with neighborhood similarity in the PPPMI embedding space, while the factor $\sqrt{\bar{f}_i \bar{f}_j}$ upweights pairs that are both related and operationally frequent without letting frequency alone dominate the score. Equation (14) plays the analogous role for LDA by weighting pairs according to their shared latent-topic mass. For an arbitrary pair-weight matrix W , the objective is

$$F_{\text{pair}}(p; W) = \frac{\sum_{i < j} W_{ij} C_{p(i)p(j)}}{\sum_{i < j} W_{ij}}. \quad (15)$$

Simulated annealing maximizes Equation (15), which is consistent with the surrogate principle of placing strongly related SKU pairs in nearby slots.

4.5.2. Neighborhood Search and Temperature Schedule

A neighborhood move swaps two slots. First, SKU i is selected according to

$$P(i) \propto f_i + w_{\text{sel}} \max_j f_j,$$

where $w_{\text{sel}} = 0.1$ smooths the selection so that low-frequency SKUs are still explored. The slot containing SKU i is then swapped with a uniformly sampled slot from the remainder. Accordingly, the proposal distribution is generated by a frequency-biased first choice

followed by uniform partner selection over the remaining slots. The goal is heuristic search rather than unbiased sampling, so frequently requested SKUs are revisited more often while every remaining slot can still appear as the swap partner. Let Δ be the difference between the new and old objective values. The acceptance probability is

$$P(\text{accept}) = \begin{cases} 1, & \Delta \geq 0, \\ \exp(\Delta/T_t), & \Delta < 0. \end{cases} \quad (16)$$

The temperature follows geometric cooling:

$$T_t = T_0 \left(\frac{T_{\text{end}}}{T_0} \right)^{t/(T_{\text{max}}-1)}, \quad (17)$$

Equations (16) and (17) define the acceptance and cooling rules used by the SA variants. where T_0 is the 80th percentile of $|\Delta|$ over pre-sampled worsening swaps and

$$T_{\text{end}} = \max(0.02T_0, 10^{-5}).$$

Here, T_0 controls the initial willingness to accept worsening swaps, T_{end} sets the minimum exploration temperature, and T_{max} is the total number of swap iterations in the schedule. This configuration is shared across all SA-based comparisons.

sa-pmi

sa-pmi uses $p^{(0)} = p_{\text{ck-pmi}}$ as the initial solution and optimizes Equation (15) with W^{PMI} . The difference between ck-pmi and sa-pmi therefore isolates the additional effect of swap-based local search when the PMI representation is fixed.

sa-lda

sa-lda uses $p^{(0)} = p_{\text{ck-lda}}$ and optimizes $F_{\text{pair}}(p; W^{\text{LDA}})$. Comparing ck-lda with sa-lda shows whether clustering alone is sufficient under the LDA representation or whether heuristic local search adds value.

lda-frequency-nearest

lda-frequency-nearest uses $p^{(0)} = p_{\text{frequency-nearest}}$ as its initial solution and then optimizes the LDA objective $F_{\text{pair}}(p; W^{\text{LDA}})$. Relative to frequency-nearest, it measures the benefit of adding LDA-driven local search on top of a classical turnover-based layout. Relative to sa-lda, it measures the effect of the initial solution when the LDA objective is held fixed.

pmi-frequency-nearest

pmi-frequency-nearest also starts from $p^{(0)} = p_{\text{frequency-nearest}}$, but it optimizes $F_{\text{pair}}(p; W^{\text{PMI}})$ instead. It is included to test whether the dependence on the initial solution observed for LDA is specific to LDA or also appears with PMI.

Intent of the Comparison Design

The comparison design in Table 2 serves three purposes. Comparison with classical baselines tests whether co-occurrence information is useful at all. Comparison between ck-pmi/ck-lda and between sa-pmi/sa-lda isolates the representational effect of PPPMI versus LDA under matched construction mechanisms. Finally, comparison with pmi-frequency-nearest and lda-frequency-nearest reveals whether performance depends on the interaction between the co-occurrence representation and the initial solution.

5. Experimental Design

5.1. Research Objective and Questions

The experiments replace only the storage assignment p while keeping the planning pipeline fixed. The purpose is to identify how differences in co-occurrence representation and placement construction affect execution performance. The central question is not whether one representation universally dominates the other, but whether PPPMI and LDA behave differently when the order structure is controlled and the only experimental factor varied is the storage assignment method. We formulate the following research questions:

1. **RQ1:** Do co-occurrence-aware placements outperform baselines that do not use co-occurrence information?
2. **RQ2:** Which co-occurrence representation is more effective, PPPMI or LDA?
3. **RQ3:** How do placement construction style and initial solution affect the observed gains?
4. **RQ4:** How stable are the above tendencies under changes in goal policy, noise rate, number of agents, order arrival rate, and warehouse layout?
5. **RQ5:** To what extent do the findings remain valid under different order structures and different numbers of LDA topics?

5.2. Compared Methods and Comparison Intent

The compared methods are random, balance, frequency-nearest, ck-pmi, ck-lda, sa-pmi, sa-lda, pmi-frequency-nearest, and lda-frequency-nearest. These methods are organized around three comparison axes:

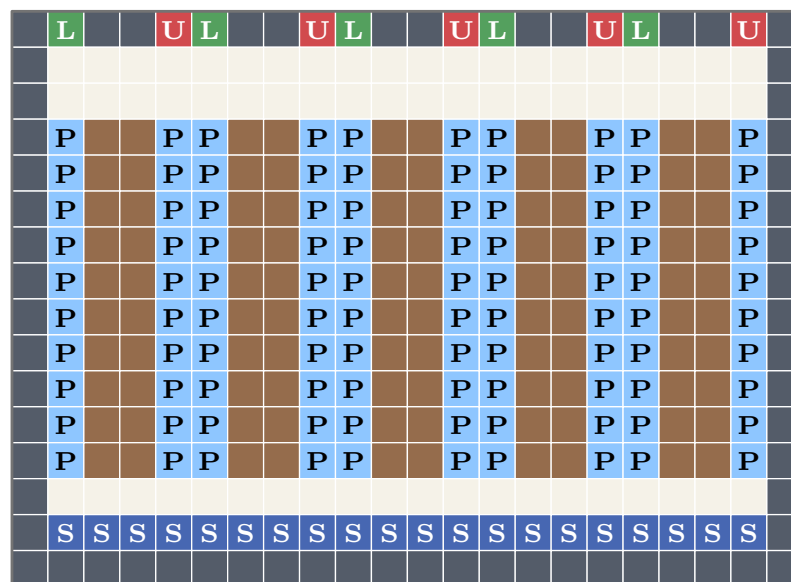
1. **Comparison against non-co-occurrence baselines:** random, balance, and frequency-nearest serve as reference points for testing whether using co-occurrence itself improves makespan.
2. **Comparison of co-occurrence representations:** ck-pmi versus ck-lda and sa-pmi versus sa-lda compare PPPMI and LDA under matched construction mechanisms.
3. **Comparison of construction style and initial solution:** ck-lda versus sa-lda evaluates the added effect of local search, and pmi-frequency-nearest and lda-frequency-nearest test how co-occurrence-based local search interacts with a frequency-based initial solution.

For LDA-based methods, we use $K \in \{16, 32\}$ topics. These two values were selected as moderate topic granularities for a 600-SKU catalog: $K = 16$ provides coarse latent families, whereas $K = 32$ allows finer subdivisions without making the representation comparison an extensive hyperparameter search. We report both values in the main results, but because this is only a two-point comparison, we treat topic-number effects conservatively and return to them only in the overall discussion. This experimental design should therefore be read as a controlled comparison, not as a claim of exhaustive benchmarking over all representation families or all real-world order distributions. The background of the study includes practical order data in which SKU co-occurrence structure is present, but those data mix several mechanisms that are difficult to disentangle directly. The topic and basket generators are used here to separate latent-topic-like and explicit family-based co-occurrence patterns so that the relative behavior of PPPMI and LDA can be interpreted more clearly. Final external validation on real warehouse logs remains future work.

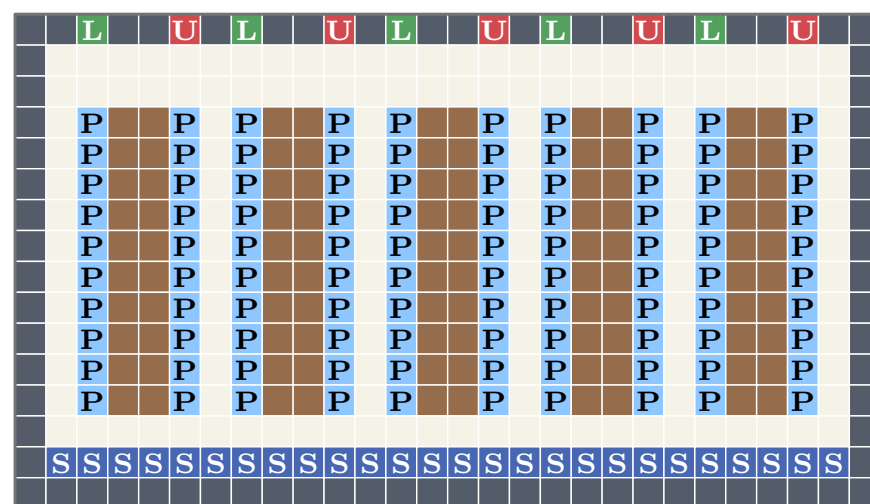
5.3. Evaluation Environment and Metrics

The evaluation fixes the agent policy π_A^{fixed} in the extended MAPD problem defined in Section 3.1 and changes only the storage assignment p . The question is therefore how the overall execution performance of the warehouse changes when only SKU placement

is altered under the same robot team and planner configuration. The primary evaluation metric is makespan, namely the completion time of the last order request among those released during the evaluation horizon. The main comparisons in the body text are based on mean makespan, computed as the arithmetic mean over completed run-level simulations for the relevant condition set. For the most important paired comparisons, Appendix A.1 additionally reports exact sign tests, bootstrap 95% confidence intervals, and paired Cohen's d_z . Table 3 summarizes the shared evaluation settings. Table 4 and Figure 2 summarize and visualize the two warehouse layouts used in the evaluation.



(a)



(b)

X blocked
 O aisle
 R rack
 P pick
 U unload
 L load
 S start/return

Figure 2. The two warehouse layouts used for evaluation. The lower legend summarizes the cell symbols. The non-well-formed map is a narrow-aisle setting with concentrated interference, whereas the well-formed map offers more room for detours and avoidance maneuvers. (a) Non-well-formed layout with narrow rack gaps and limited cross-aisle freedom. (b) Well-formed layout with wider horizontal aisles and more room for avoidance.

Table 3. Shared settings in the evaluation environment.

Item	Setting
Warehouse layouts	non-well-formed, well-formed
Storage access locations	100 in both maps
Outbound locations	5 in both maps
Shelf levels	6
Number of SKUs N	600
Order generator	latent-topic orders (topic) in the main experiment; direct-co-occurrence orders (basket) in the supplement
Topic concentration α_{topic}	0.3 in the main topic experiment
Training orders	10,000 orders
Evaluation horizon	300 timesteps
Order arrival rate	0.10, 0.20
Order size	1–10 SKUs per order
Goal policy	random, nearest
LDA training iterations	adaptive 2400–3600 Gibbs sweeps
Simulated annealing iterations	adaptive 6000–9000 swap proposals
Maximum simulation length	10,000 timesteps
Orientation-aware motion	enabled
Rotation cost	3
Service times	fixed at 5 for picking, unloading, and loading

Table 4. Structural comparison of the evaluation maps.

Layout	Grid	#O	#P	#R	#U	#L	#S	Structural Characteristic
non-well-formed	16×22	60	100	100	5	5	20	Narrow aisles; interference concentrated in shared corridors
well-formed	16×28	138	100	100	5	5	26	Wide aisles; more room for detours and avoidance

Both maps have 100 storage access cells. With six shelf levels, this yields 600 storage slots in total. Using both layouts lets us test whether the effect of a placement method is tied to one specific geometry.

5.4. Main Experiment: Orders with Latent Topic Structure

The main experiment uses orders with latent topic structure in both training and evaluation (topic $\rightarrow$ topic). This is the main test bed for RQ1–RQ4 because it directly examines whether representational differences propagate to makespan when the order stream contains latent topic structure. The purpose of this experiment is not to claim a universal advantage for LDA. Rather, it is to test whether an order stream that truly contains latent mixture structure creates a controlled regime in which LDA is better matched than PPPMI under the fixed execution pipeline.

The varying factors are two maps, two agent counts, two order rates, three noise levels, two goal policies, five training seeds, and ten test seeds. This supports both overall averages across all conditions and condition-wise summaries for goal policy, noise, load, and layout. The most important comparisons are ck-pmi versus ck-lda, sa-pmi versus sa-lda, ck-lda versus sa-lda, and the interaction with the frequency-nearest initial solution through pmi-frequency-nearest and lda-frequency-nearest.

5.5. Supplementary Experiment: Robustness to Order Structure

As a supplementary experiment, we evaluate orders with direct co-occurrence structure in both training and evaluation (basket $\rightarrow$ basket), where the generator creates strong within-family co-occurrence. This experiment tests whether the main findings persist under an order structure different from the latent-topic setting. Its role is to check whether the main result is merely generator specific or whether some conclusions survive when the

order structure is better matched to direct pairwise statistics. The full results are reported in Appendix A.4; the main text discusses only the high-level implications in Section 6.6.

5.6. Supplementary Diagnostic: Correspondence Between the Surrogate and Makespan

Section 4.1 introduced a surrogate based on weighted SKU-pair closeness. As a diagnostic, we examine how well this surrogate aligns with actual makespan in the simulator. The aim is not to re-optimize each method but to visualize whether the design principle of placing co-occurring SKUs nearby corresponds monotonically to execution performance.

For this diagnostic, we fix goal policy=random, order rate 0.20, and noise 0.0, and consider the four combinations of non-well-formed/well-formed and 6/12 agents. For each condition, we use the five training-seed placements generated by the representative methods random, balance, frequency-nearest, ck-pmi, sa-pmi, ck-lda ($K = 16, 32$), and sa-lda ($K = 16, 32$), for a total of 45 points per panel. The horizontal axis is a unified F_{pair} computed from a common W_{LDA} with $K = 16$, reconstructed from the same training corpus for all placements. The vertical axis is the mean makespan across the ten test seeds for that placement.

6. Results and Discussion

This section answers the research questions in order. Section 6.1 uses Table 5 to establish the overall main-experiment ranking for RQ1, Sections 6.2 and 6.3 unpack the representation and construction effects for RQ2–RQ3, and Section 6.4 uses Table 6 to show condition dependence for RQ4. Section 6.5 then uses Figure 3 as a diagnostic for the surrogate in Section 4.1, and Section 6.6 summarizes the robustness check under direct-co-occurrence orders (basket) from Appendix A.4 for RQ5.

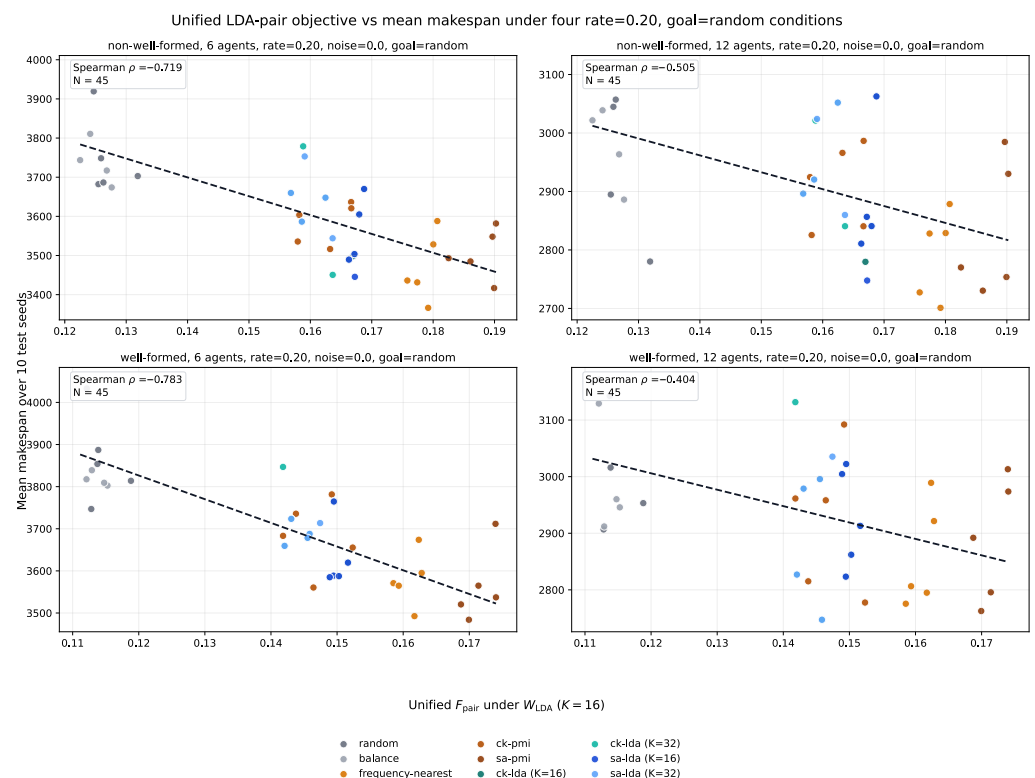


Figure 3. Unified surrogate F_{pair} built from W_{LDA} ($K = 16$) versus mean makespan for the four goal policy = random, rate=0.20, noise=0.0 conditions (non-well-formed/well-formed $\times$ 6/12 agents). Each point represents one training-seed placement, and the vertical axis reports the mean makespan over ten test seeds.

Table 5. Overall mean makespan in the main experiment.

Method	Makespan	Improvement [%]
random	2757.79	0.00
balance	2765.83	−0.29
frequency-nearest	2718.84	1.41
ck-pmi	2716.80	1.49
sa-pmi	2740.65	0.62
pmi-frequency-nearest	2800.86	−1.56
ck-lda ($K = 16$)	2633.16	4.52
ck-lda ($K = 32$)	2646.02	4.05
sa-lda ($K = 16$)	2630.97	4.60
sa-lda ($K = 32$)	2649.42	3.93
lda-frequency-nearest ($K = 16$)	2629.98	4.63
lda-frequency-nearest ($K = 32$)	2627.44	4.73

Table 6. Improvement rates by operating condition in the main experiment (relative to random under the same condition). For readability, we report ck-lda and sa-lda with $K = 16$ and lda-frequency-nearest with $K = 32$.

Factor	Condition	Frequency-Nearest	ck-pmi	sa-pmi	pmi-Frequency-Nearest	ck-lda ($K = 16$)	sa-lda ($K = 16$)	lda-Frequency-Nearest ($K = 32$)
goal policy	random	6.53	3.19	5.71	6.37	4.19	4.32	7.47
	nearest	−3.40	−0.11	−4.17	−9.01	4.82	4.86	2.15
noise	0.0	1.80	1.34	1.32	−1.61	6.04	6.14	5.48
	0.1	0.84	2.14	0.16	−2.11	3.61	3.61	4.59
	0.2	1.59	0.98	0.37	−0.95	3.88	4.03	4.09
agents	6	2.51	1.85	1.99	0.29	4.36	4.44	5.47
	12	0.07	1.03	−1.04	−3.81	4.68	4.77	3.81
order rate	0.10	2.33	1.74	1.90	−0.57	4.23	4.28	5.35
	0.20	0.87	1.32	−0.16	−2.13	4.55	4.65	4.36
layout	non-well-formed	1.59	1.19	0.99	−1.31	4.40	4.50	5.26
	well-formed	1.24	1.78	0.26	−1.81	4.64	4.69	4.21

Unless otherwise stated, all values in the main-text tables are mean makespans in the main experiment, averaged over completed run-level simulations. Main-table makespans and improvement rates are reported to two decimal places for consistency. The overall averages aggregate across all combinations of two maps, two order rates, three noise levels, two goal policies, two agent counts, five training seeds, and ten test seeds. Appendix A.1 Table A1 reports exact sign tests for the main comparisons, Table A2 reports descriptive statistics, and Table A3 reports bootstrap confidence intervals and paired effect sizes.

6.1. RQ1: Overall Results in the Main Experiment

This subsection uses the overall averages in Table 5 to establish the aggregate ranking and answer RQ1.

6.1.1. Question Addressed in This Subsection

RQ1 asks whether co-occurrence-aware placements improve performance relative to baselines without co-occurrence information.

6.1.2. Results

Table 5 shows that the lowest overall mean makespan was achieved by lda-frequency-nearest with $K = 32$. lda-frequency-nearest with $K = 16$, sa-lda with $K = 16$, and ck-lda with $K = 16$ followed very closely, so the top four methods in

the full main-experiment average were all LDA-based. In contrast, frequency-nearest and ck-pmi improved on random only slightly, balance did not improve at all, and pmi-frequency-nearest was worse than random. For the headline comparison, lda-frequency-nearest ($K = 32$) beats random by 130.01 makespan units on the paired runs, with an exact sign-test p -value of 1.0×10^{-107} , a bootstrap 95% confidence interval of [117.15, 142.58], and paired Cohen's $d_z = 0.412$.

6.1.3. Statistical Supplement

Appendix A.1 Table A1 shows that the difference between lda-frequency-nearest ($K = 32$) and random is also supported by run-level sign comparison. Appendix A.2 Table A2 further indicates that the variance of lda-frequency-nearest ($K = 32$) is slightly smaller than that of random. Appendix A.3 Table A3 reports a mean difference of 130.01 with a bootstrap 95% confidence interval of [117.15, 142.58] and paired Cohen's $d_z = 0.412$.

6.1.4. Interpretation

Taken over all conditions, the results indicate that co-occurrence-aware storage assignment can improve warehouse execution, but the effect is not automatic for every co-occurrence-based method. In the topic-structured main experiment, the improvement is primarily driven by LDA-based methods. Under the present setting, the outcome depends less on whether co-occurrence is used at all and more on which representation and construction mechanism are combined.

6.2. RQ2: Effect of the Co-Occurrence Representation

This subsection uses the matched PPPMI-versus-LDA pairs in Tables 5 and 6 to isolate the representation effect.

6.2.1. Question Addressed in This Subsection

RQ2 asks whether PPPMI or LDA is more effective when the construction mechanism is held fixed.

6.2.2. Results

In Table 5, ck-lda ($K = 16$) is 83.64 points lower in the aggregate mean makespan than ck-pmi, corresponding to a relative advantage of 3.08%. The paired run-level analysis gives a mean difference of 80.26, an exact sign-test p -value of 1.8×10^{-19} , a bootstrap 95% confidence interval of [67.31, 93.25], and $d_z = 0.249$ for this comparison. Likewise, sa-lda ($K = 16$) is 109.68 points lower than sa-pmi in the aggregate table, a relative advantage of 4.00%; the paired mean difference is 108.38, with sign-test $p = 2.8 \times 10^{-12}$, 95% CI [94.43, 122.58], and $d_z = 0.302$. The condition-wise summary in Table 6 further shows that the representative LDA-based methods outperform random in every reported factor-wise aggregation, whereas PMI-based methods can lose their advantage under goal policy = nearest or under heavier operating conditions.

6.2.3. Statistical Supplement

Appendix A.1 Table A1 supports both the ck-lda versus ck-pmi and the sa-lda versus sa-pmi comparisons. The corresponding confidence intervals in Appendix A.3 Table A3 are [67.31, 93.25] and [94.43, 122.58], respectively, with paired Cohen's d_z values of 0.249 and 0.302.

6.2.4. Interpretation

The correct interpretation is conditional rather than universal. In the main experiment with latent topic structure in both training and evaluation (topic→topic), the order stream contains latent mixture structure by construction, so LDA is the better matched representation for that regime. PMI faithfully reflects observed pairwise co-occurrence, but LDA can express relationships among SKUs even when direct co-occurrence is sparse by tying them together through shared latent topics. This is therefore evidence of representational fit under topic-structured orders, not evidence that LDA must dominate for every order generator. Section 6.6 revisits the question under a different generator.

6.3. RQ3: Effect of Construction Method and Initial Solution

This subsection uses Table 5 together with the factor-wise patterns in Table 6 to separate the effect of local search from the effect of the initial solution.

6.3.1. Question Addressed in This Subsection

RQ3 asks how the addition of local search and the choice of initial solution affect performance.

6.3.2. Results

Table 5 shows that the difference between sa-lda ($K = 16$) and ck-lda ($K = 16$) is only 2.19, so the average effect of adding local search is limited in the full aggregate. The paired statistics are also small for this contrast, with sign-test $p = 0.107$, 95% CI [0.39, 4.00], and $d_z = 0.048$. By contrast, lda-frequency-nearest is the best method overall among the LDA family, whereas pmi-frequency-nearest is 82.02 points worse than frequency-nearest; for the latter comparison, the paired analysis gives sign-test $p = 1.1 \times 10^{-13}$, 95% CI [−95.27, −67.38], and $d_z = -0.236$. Moreover, sa-pmi does not improve over ck-pmi. Table 6 reveals a more detailed pattern: lda-frequency-nearest is especially strong under goal policy=random, with 6 agents, order rate 0.10, and the non-well-formed layout, while ck-lda/sa-lda are relatively stronger under goal policy=nearest, with 12 agents, order rate 0.20, and the well-formed layout.

6.3.3. Statistical Supplement

Appendix A.1 Table A1 shows that the difference between sa-lda ($K = 16$) and ck-lda ($K = 16$) is not decisive ($p = 0.107$). In contrast, the worsening of pmi-frequency-nearest relative to frequency-nearest is clearly supported. Appendix A.3 Table A3 confirms that the mean difference between sa-lda ($K = 16$) and ck-lda ($K = 16$) is small, while the mean difference for pmi-frequency-nearest versus frequency-nearest is negative with a nontrivial confidence interval.

6.3.4. Interpretation

These results show that adding local search is not uniformly beneficial. Compatibility between the co-occurrence representation and the initial solution matters. In particular, LDA benefits from combining a frequency-based initial layout with local search, whereas the analogous PMI configuration does not. Under our setting, it is therefore more appropriate to discuss performance as a joint property of representation, initial solution, and search mechanism than to rank construction styles in isolation.

6.4. RQ4: Sensitivity to Operating Conditions

This subsection reads Table 6 factor by factor to show which findings remain stable across operating conditions and which depend on the regime.

6.4.1. Question Addressed in This Subsection

RQ4 asks whether the main tendencies persist when the goal policy, noise rate, number of agents, order arrival rate, and layout are varied.

6.4.2. Results

Table 6 shows that under goal policy=random, even frequency-nearest and pmi-frequency-nearest can improve substantially, whereas under goal policy=nearest, many frequency-based and PMI-based methods lose their advantage. By contrast, the representative LDA-based methods outperform random in every factor-wise aggregation reported in the table. In particular, under goal policy=nearest, ck-lda ($K = 16$) and sa-lda ($K = 16$) still improve by 4.82–4.86%. Even when the noise level is increased from 0.0 to 0.2, the improvement of the LDA-based methods stays in the range 3.61–6.14%. At the same time, the best-performing LDA-based method changes with the condition: lda-frequency-nearest is stronger in lighter regimes and under goal policy=random, while ck-lda/sa-lda are stronger in heavier conditions and under goal policy=nearest.

6.4.3. Interpretation

As a representation family, LDA appears comparatively robust to changes in operating conditions in the main experiment, although this statement is conditional on the synthetic topic-structured regime. However, the best configuration within the LDA family is not universal. Changes in goal assignment and congestion alter which initial solution and search structure are most favorable. The evidence in this subsection is descriptive rather than inferential, because Table 6 aggregates over factor-wise slices rather than paired run-by-run contrasts. The correct conclusion is therefore not that one single LDA-based method dominates all regimes, but that within topic-structured orders LDA is a promising feature family across diverse operating conditions.

6.5. Diagnostic: Correspondence Between the Surrogate Objective and Makespan

This subsection uses Figure 3 only as a diagnostic; it is not another method ranking, but a check of whether the surrogate from Section 4.1 is directionally aligned with measured makespan.

6.5.1. Question Addressed in This Subsection

This diagnostic asks whether the surrogate from Section 4.1 behaves as a meaningful design indicator for makespan, at least under representative operating conditions.

6.5.2. Results

Figure 3 shows a negative Spearman correlation between F_{pair} and mean makespan in all four conditions. The correlation values are -0.719 for non-well-formed/6 agents, -0.505 for non-well-formed/12 agents, -0.783 for well-formed/6 agents, and -0.404 for well-formed/12 agents. Thus, under goal policy = random and noise 0.0, placements ranked higher by the unified surrogate tend also to yield smaller mean makespans.

6.5.3. Interpretation

The diagnostic does not validate Equation (5) as a theorem, but it does support the directional mechanism proposed in Section 4.1. Placements that bring related SKUs closer tend also to yield smaller mean makespans under representative conditions, which is consistent with lower within-order travel and shorter occupancy windows around frequently visited picks. The relationship is especially strong with 6 agents, suggesting that this mechanism is visible when multi-agent interference is moderate. The weaker correlation with 12 agents indicates that stronger multi-agent interference reduces how

much of the makespan can be explained by a placement-only surrogate. Accordingly, F_{pair} should be interpreted not as a precise predictor of makespan but as a useful design heuristic.

6.6. RQ5: Robustness to Order Structure

This subsection uses Appendix A.4 Table A4 to leave the main latent-topic setting and test whether the qualitative conclusions survive under orders with direct co-occurrence structure (basket).

6.6.1. Question Addressed in This Subsection

The first half of RQ5 asks whether the main findings persist under an order structure different from the latent-topic generator.

6.6.2. Results

The full results for direct-co-occurrence orders in both training and evaluation (basket $\rightarrow$ basket) are reported in Appendix A.4. Table A4 shows that under the plain CK-style comparison, ck-pmi outperforms ck-lda. At the same time, lda-frequency-nearest ($K = 32$) is still the best overall method, with a mean makespan of 2280.15.

6.6.3. Interpretation

This supplementary experiment directly addresses the concern that the main result might only reflect alignment with the latent-structure generator (topic). When direct family-based co-occurrence is very strong, PPPMI can be competitive in the plain representation comparison. At the same time, LDA-driven search on top of a frequency-based initial layout remains effective. The main conclusion is therefore not that LDA uniformly dominates PMI, but that the favorable comparison axis depends on the order structure. The interaction between LDA and the initial solution remains important even outside the latent-topic setting.

6.7. Overall Discussion

6.7.1. Answers to the Research Questions

For RQ1, the main-experiment average shows that co-occurrence-aware placement can improve makespan over classical baselines, but the improvement does not appear uniformly for every co-occurrence-based method. For RQ2, LDA was better matched than PPPMI in the main experiment where both training and evaluation orders contain latent topic structure (topic $\rightarrow$ topic), but the direct-co-occurrence check (basket) shows that this advantage is conditional on the order structure. For RQ3, compatibility between the representation and the initial solution matters more than a simple ranking of clustering versus local search. For RQ4, LDA-based methods were comparatively stable across changes in operating conditions within the topic-structured synthetic regime, although the best LDA-based configuration still changed with the regime. For RQ5, changing the order structure changes which comparison axis is favorable; topic number is observed only through a two-point comparison and should therefore be interpreted cautiously.

6.7.2. Supplementary Observation on the Number of Topics

Within the LDA family, ck-lda and sa-lda both perform slightly better at $K = 16$ than at $K = 32$, whereas lda-frequency-nearest is slightly better at $K = 32$. The differences are modest and are based only on two topic counts. It is therefore more appropriate to regard topic number as part of the overall configuration rather than to draw strong conclusions about its isolated sensitivity.

6.7.3. Why Did LDA Help in the Main Experiment?

In the main experiment, the order generator intentionally contains latent topic structure so that the representational comparison is identifiable. Under that controlled regime, LDA is better matched because it can smooth co-occurrence over several SKUs and capture relations that are not always visible in direct pair counts. By contrast, PPPMI is more literal and may be better matched when direct co-occurrence within product groups dominates, as in the basket condition. This reinforces that the choice of representation should be interpreted as a question of fit to the order structure rather than as a universal ranking for real warehouse data.

6.7.4. Limitations

This study has at least four limitations. First, the evaluation relies on synthetic latent-structure and direct-co-occurrence order generators (topic and basket), so external validation on real warehouse logs remains incomplete. This does not mean that the study was designed in isolation from practice: the broader research program, including the conference-stage work that motivated this journal extension, is grounded in real order data where SKU co-occurrence structure is present. However, real logs mix latent-topic-like co-occurrence, explicit family-level co-occurrence, noise, and demand variation. The topic and basket generators were therefore used as controlled abstractions to separate those effects and make the PPPMI-versus-LDA comparison interpretable. They help characterize when each representation is better matched, but they do not replace final validation on real warehouse logs. In particular, real logs may combine latent-topic-like relations, explicit family-level associations, substitutions, seasonality, promotions, and operational batching rules in proportions that neither synthetic generator exactly matches. Second, the planner and goal policy are fixed to a small set of design choices, and the results may depend on those choices. Third, although Appendix A.1 reports exact sign tests, bootstrap confidence intervals, and paired effect sizes for the main comparisons, the main-text discussion still centers on mean makespan. More extensive statistical analysis over condition-wise slices remains future work. Fourth, the paper focuses on offline placement optimization and does not compare the computational cost of LDA training, PPPMI construction, simulated annealing, and broader alternative representation families such as NMF, CTM, or BERT-based embeddings in detail.

7. Conclusions

This paper evaluated storage location assignment in automated warehouses as an extended MAPD problem under a fixed execution pipeline and compared PPPMI and LDA as two deliberately chosen representation families: direct pairwise co-occurrence versus latent-topic smoothing. In the controlled topic regime, lda-frequency-nearest ($K = 32$) achieved the best overall mean makespan, and the LDA-based methods tended to compare favorably on average. The supplementary basket condition showed that PPPMI can be competitive when explicit pairwise structure is stronger, while LDA-driven local search on top of a frequency-based initial layout remained strong. The main takeaway is therefore not that LDA is universally superior, but that PPPMI and LDA are better matched to different order structures under the tested execution pipeline. The synthetic generators should be read as controlled abstractions motivated by co-occurrence structure observed in real order data, not as a substitute for final external validation. Future work should compare additional representation families such as NMF, CTM, and BERT-based embeddings, and should test the resulting methods on real warehouse logs under rolling-horizon operation.

Author Contributions: T.I. performed conceptualization, methodology, software, validation, formal analysis, investigation, and writing of the original draft. T.H. contributed the experimental simulator used in the study and supported its use in the experiments. N.H. contributed to methodology refinement, interpretation of results, and review and editing. H.S. and I.N. contributed to interpretation of results and review and editing. H.S. and I.N. supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source code, generated data, and manuscript materials used in this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: Naoki Hattori is employed by Toyota Industries Corporation. The remaining authors declare no conflict of interest. This study was conducted as a collaborative research project among the participating institutions, and the authors do not consider that collaborative relationship itself constitutes a conflict of interest. No other potential commercial interests are declared.

Appendix A. Supplementary Results

Appendix A.1. Statistical Supplement for the Main Comparisons

This appendix reports exact sign tests, bootstrap 95% confidence intervals, and effect sizes for the main paired makespan comparisons. In Table A1, the reported mean difference is “makespan of method 2 minus makespan of method 1”, so a positive value means that method 1 performs better on average.

Table A1. Exact sign tests for the main pairwise comparisons.

Comparison	Wins–Losses–Ties	Mean Difference	<i>p</i> -Value
lda-frequency-nearest ($K = 32$) vs. random	1730–668–1	130.0	1.0×10^{-107}
ck-lda ($K = 16$) vs. ck-pmi	1416–975–3	80.3	1.8×10^{-19}
sa-lda ($K = 16$) vs. sa-pmi	1365–1023–3	108.4	2.8×10^{-12}
sa-lda ($K = 16$) vs. ck-lda ($K = 16$)	114–90–2190	2.2	0.107
pmi-frequency-nearest vs. frequency-nearest	1016–1380–3	–81.4	1.1×10^{-13}

Table A1 confirms that the improvements of lda-frequency-nearest ($K = 32$) over random, ck-lda ($K = 16$) over ck-pmi, and sa-lda ($K = 16$) over sa-pmi are supported even at the run level. By contrast, the difference between sa-lda ($K = 16$) and ck-lda ($K = 16$) is small, and pmi-frequency-nearest is consistently worse than frequency-nearest.

Appendix A.2. Descriptive Statistics for the Overall Main-Experiment Average

Table A2 reports distributional statistics for the overall main-experiment averages. The standard deviation is the sample standard deviation, and the IQR is the difference between the 75th and 25th percentiles.

Table A2. Descriptive statistics for the overall main-experiment averages.

Method	Mean	Std. Dev.	Median	IQR
random	2757.79	960.49	2672.00	1602.50
balance	2765.83	966.74	2668.00	1632.00
frequency-nearest	2718.84	992.48	2588.00	1599.50
ck-pmi	2716.80	958.68	2610.00	1576.50
sa-pmi	2740.65	1006.31	2622.00	1645.00
pmi-frequency-nearest	2800.86	1042.81	2656.00	1685.25
ck-lda ($K = 16$)	2633.16	919.22	2561.50	1508.25
ck-lda ($K = 32$)	2646.02	927.94	2557.00	1526.00
sa-lda ($K = 16$)	2630.97	918.04	2562.00	1510.25
sa-lda ($K = 32$)	2649.42	926.26	2542.00	1531.00

Table A2. *Cont.*

Method	Mean	Std. Dev.	Median	IQR
lda-frequency-nearest ($K = 16$)	2629.98	938.92	2529.00	1541.75
lda-frequency-nearest ($K = 32$)	2627.44	941.22	2521.00	1541.75

Appendix A.3. Confidence Intervals and Effect Sizes

Table A3 gives paired bootstrap 95% confidence intervals and paired Cohen's d_z for the same main comparisons as Table A1. The bootstrap uses 10,000 resamples of the paired differences.

Table A3. Bootstrap confidence intervals and effect sizes for the main pairwise comparisons.

Comparison	Mean Difference	95% CI	d_z
lda-frequency-nearest ($K = 32$) vs. random	130.01	[117.15, 142.58]	0.412
ck-lda ($K = 16$) vs. ck-pmi	80.26	[67.31, 93.25]	0.249
sa-lda ($K = 16$) vs. sa-pmi	108.38	[94.43, 122.58]	0.302
sa-lda ($K = 16$) vs. ck-lda ($K = 16$)	2.19	[0.39, 4.00]	0.048
pmi-frequency-nearest vs. frequency-nearest	−81.37	[−95.27, −67.38]	−0.236

Appendix A.4. Order-Data Ablation

This appendix summarizes the condition with direct-co-occurrence orders in both training and evaluation (basket→basket), where the generator creates strong within-family correlation with weak inter-family mixing. Compared with the main experiment using latent-structure orders in both training and evaluation (topic→topic), this condition emphasizes explicit family-based co-occurrence rather than latent topic mixtures.

Appendix A.4.1. Generation of Family-Based Orders

Each order first chooses a primary family, namely the main product group around which the order is formed, optionally adds a secondary family with probability p_{mix} , and then samples items without replacement with decreasing weights for the primary-family core items (central representatives), halo items (related but less central products), and secondary-family items. This corresponds to a setting where direct family co-occurrence is more dominant than latent topic mixtures.

Appendix A.4.2. Experimental Setting

The same two maps, two agent counts, two load levels, three noise rates, two goal policies, five training seeds, and ten test seeds are used as in the main experiment. Service times are again fixed at 5 for picking, unloading, and loading. The compared methods are random, frequency-nearest, ck-pmi, ck-lda, sa-lda, and lda-frequency-nearest.

Table A4 shows that under direct-co-occurrence orders in both training and evaluation (basket→basket), ck-pmi outperforms ck-lda in the plain CK-style comparison, but lda-frequency-nearest ($K = 32$) is still the best method overall. Thus, the combination of a frequency-based initial solution and LDA-driven search remains effective even when direct family structure is strong.

Table A4. Representative methods under the basket→basket condition.

Method	Mean Makespan	Improvement over Random [%]
random	2491.14	0.00
frequency-nearest	2469.04	0.89
ck-pmi	2356.85	5.39

Table A4. Cont.

Method	Mean Makespan	Improvement over Random [%]
ck-lda ($K = 16$)	2383.01	4.34
sa-lda ($K = 32$)	2369.27	4.89
lda-frequency-nearest ($K = 16$)	2286.17	8.23
lda-frequency-nearest ($K = 32$)	2280.15	8.47

References

- De Koster, R.; Le-Duc, T.; Roodbergen, K.J. Design and control of warehouse order picking: A literature review. *Eur. J. Oper. Res.* **2007**, *182*, 481–501. [\[CrossRef\]](#)
- Gu, J.; Goetschalckx, M.; McGinnis, L.F. Research on warehouse operation: A comprehensive review. *Eur. J. Oper. Res.* **2007**, *177*, 1–21. [\[CrossRef\]](#)
- Wurman, P.R.; D’Andrea, R.; Mountz, M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Mag.* **2008**, *29*, 9.
- Allgor, R.; Cezik, T.; Chen, D. Algorithm for Robotic Picking in Amazon Fulfillment Centers Enables Humans and Robots to Work Together Effectively. *INFORMS J. Appl. Anal.* **2023**, *53*, 266–282. [\[CrossRef\]](#)
- Barnhart, C.; Jacquillat, A.; Schmid, A. Robotic Warehousing Operations: A Learn-Then-Optimize Approach to Large-Scale Neighborhood Search. *INFORMS J. Optim.* **2024**, *7*, 171–194. [\[CrossRef\]](#)
- Li, J.; Tinka, A.; Kiesel, S.; Durham, J.W.; Kumar, T.S.; Koenig, S. Lifelong multi-agent path finding in large-scale warehouses. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 11272–11281.
- Ma, H.; Li, J.; Kumar, T.; Koenig, S. Lifelong multi-agent path finding for online pickup and delivery tasks. *arXiv* **2017**, arXiv:1705.10868. [\[CrossRef\]](#)
- Liu, M.; Ma, H.; Li, J.; Koenig, S. Task and path planning for multi-agent pickup and delivery. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), Montreal, QC, Canada, 13–17 July 2019.
- Salzman, O.; Stern, R. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, Auckland, New Zealand, 9–13 May 2020; pp. 1711–1715.
- Frazelle, E.H. *Stock Location Assignment and Order Picking Productivity*; Georgia Institute of Technology: Atlanta, GA, USA, 1989.
- Reyes, J.; Solano-Charis, E.; Montoya-Torres, J. The storage location assignment problem: A literature review. *Int. J. Ind. Eng. Comput.* **2019**, *10*, 199–224. [\[CrossRef\]](#)
- Islam, M.S.; Uddin, M.K. Correlated storage assignment approach in warehouses: A systematic literature review. *J. Ind. Eng. Manag.* **2023**, *16*, 294–318. [\[CrossRef\]](#)
- Zhang, R.Q.; Wang, M.; Pan, X. New model of the storage location assignment problem considering demand correlation pattern. *Comput. Ind. Eng.* **2019**, *129*, 210–219. [\[CrossRef\]](#)
- Hattori, N.; Kondo, T.; Nawa, M.; Kato, N.; Noda, I. Semi-Optimal Storage Location Assignment in Automated Warehouses via Constrained Clustering with Co-Occurrence Information. In *Proceedings of the 38th Annual Conference of the Japanese Society for Artificial Intelligence*; Japanese Society for Artificial Intelligence, Shizuoka, Japan, 2024; p. 3Xin204.
- Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
- Griffiths, T.L.; Steyvers, M. Finding scientific topics. *Proc. Natl. Acad. Sci. USA* **2004**, *101*, 5228–5235. [\[CrossRef\]](#) [\[PubMed\]](#)
- Lee, D.D.; Seung, H.S. Learning the Parts of Objects by Non-Negative Matrix Factorization. *Nature* **1999**, *401*, 788–791. [\[CrossRef\]](#)
- Blei, D.M.; Lafferty, J.D. A Correlated Topic Model of Science. *Ann. Appl. Stat.* **2007**, *1*, 17–35. [\[CrossRef\]](#)
- Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 3–5 June 2019; pp. 4171–4186. [\[CrossRef\]](#)
- Ito, T.; Hattori, N.; Sakaji, H.; Noda, I. LDAS: Proposal for Semi-optimal Storage Locations Using LDA and SA. In *Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems*; Springer: Cham, Switzerland, 2025; pp. 191–202.
- Hirayama, T.; Yoshida, K.; Sakaji, H.; Noda, I. SHARP: A Complete Multi-Task MAPD Planner for Realistic Warehouse Scenarios. In Proceedings of the 18th International Conference on Agents and Artificial Intelligence (ICAART 2026), Marbella, Spain, 5–7 March 2026; Volume 1, pp. 648–659. [\[CrossRef\]](#)
- Kudo, F.; Cai, K. Anytime multi-task multi-agent pickup and delivery under energy constraint. *IEEE Robot. Autom. Lett.* **2024**, *9*, 10145–10152. [\[CrossRef\]](#)

23. Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In Proceedings of the International Symposium on Combinatorial Search, Napa, CA, USA, 16–17 July 2019; Volume 10, pp. 151–158.
24. Zhang, Y.; Jiang, H.; Bhatt, V.; Nikolaidis, S.; Li, J. Guidance graph optimization for lifelong multi-agent path finding. *arXiv* **2024**, arXiv:2402.01446. [[CrossRef](#)]
25. Zang, H.; Zhang, Y.; Jiang, H.; Chen, Z.; Harabor, D.; Stuckey, P.J.; Li, J. Online guidance graph optimization for lifelong multi-agent path finding. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 14726–14735.
26. Muppani, V.R.; Adil, G.K. Efficient formation of storage classes for warehouse storage location assignment: A simulated annealing approach. *Omega* **2008**, *36*, 609–618. [[CrossRef](#)]
27. Lee, I.G.; Chung, S.H.; Yoon, S.W. Two-stage storage assignment to minimize travel time and congestion for warehouse order picking operations. *Comput. Ind. Eng.* **2020**, *139*, 106129. [[CrossRef](#)]
28. Kim, J.; Méndez, F.; Jimenez, J. Storage location assignment heuristics based on slot selection and frequent itemset grouping for large distribution centers. *IEEE Access* **2020**, *8*, 189025–189035. [[CrossRef](#)]
29. Church, K.; Hanks, P. Word association norms, mutual information, and lexicography. *Comput. Linguist.* **1990**, *16*, 22–29.
30. Agrawal, R.; Srikant, R. Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, 12–15 September 1994; pp. 487–499.
31. Levy, O.; Goldberg, Y. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27, pp. 2177–2185.
32. Hruschka, H. Linking multi-category purchases to latent activities of shoppers: Analysing market baskets by topic models. *Mark. ZFP–J. Res. Manag.* **2014**, *36*, 267–273. [[CrossRef](#)]
33. Reisenbichler, M.; Reutterer, T. Topic modeling in marketing: Recent advances and research opportunities: M. Reisenbichler, T. Reutterer. *J. Bus. Econ.* **2019**, *89*, 327–356.
34. Vega Carrasco, M.; Manolopoulou, I.; O’Sullivan, J.; Prior, R.; Musolesi, M. Posterior summaries of grocery retail topic models: Evaluation, interpretability and credibility. *J. R. Stat. Soc. Ser. C Appl. Stat.* **2022**, *71*, 562–588. [[CrossRef](#)]
35. Čáp, M.; Vokřínek, J.; Kleiner, A. Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures. In Proceedings of the International Conference on Automated Planning and Scheduling, Jerusalem, Israel, 7–11 June 2015; Volume 25, pp. 324–332.
36. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [[CrossRef](#)]
37. Phillips, M.; Likhachev, M. Sipp: Safe interval path planning for dynamic environments. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 5628–5635.
38. Levy, O.; Goldberg, Y.; Dagan, I. Improving distributional similarity with lessons learned from word embeddings. *Trans. Assoc. Comput. Linguist.* **2015**, *3*, 211–225. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.