2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

On-Demand Multi-Agent Basket Picking for Shopping Stores

Mattias Tiger[1,*] David Bergström[1] Simon Wijk Stranius[1,2] Evelina Holmgren[1]
Daniel de Leng[1] Fredrik Heintz[1]

***Abstract*— Imagine placing an online order on your way to the grocery store, then being able to pick the collected basket upon arrival or shortly after. Likewise, imagine placing any online retail order, made ready for pickup in minutes instead of days. In order to realize such a low-latency automatic warehouse logistics system, solvers must be made to be basket-aware. That is, it is more important that the full order (the basket) is picked timely and fast, than that any single item in the order is picked quickly. Current state-of-the-art methods are not basket-aware. Nor are they optimized for a positive customer experience, that is; to prioritize customers based on queue place and the difficulty associated with picking their order. An example of the latter is that it is preferable to prioritize a customer ordering a pack of diapers over a customer shopping a larger order, but only as long as the second customer has not already been waiting for too long. In this work we formalize the problem outlined, propose a new method that significantly outperforms the state-of-the-art, and present a new realistic simulated benchmark. The proposed method is demonstrated to work in an on-line and real-time setting, and to solve the on-demand multi-agent basket picking problem for automated shopping stores under realistic conditions.**

I. INTRODUCTION

The customer experience in grocery stores has been virtually unchanged for decades. Customers walk in, pick up their products, and pay. Only recently, technology has been utilized to unlock the ability to have self-checkout registers, entirely unmanned stores, and online checkouts. These applications reduce costs mainly by reducing labor. In any case, humans are still picking (and restocking) the products, whether it be the customer or the employees. The next step towards automating grocery stores is to introduce robot-driven order picking to satisfy customer needs (see Figure 1). By offloading these logistical tasks onto robots, the design and layout of grocery stores can be changed, which can improve both energy consumption and space occupation. Such stores can provide a greater variety of goods and keep a larger stock than unmanned stores today, with a much smaller spatial and environmental footprint.

An automated shopping store needs to incorporate product pickup and delivery using multiple agents so that customers receive their orders in a time-efficient way. In order to achieve high customer satisfaction [1], the solution must 1) take into account the completion of whole shopping carts; and 2) that the order of completion is experienced to be reasonably fair by the customers [2]. No customer wants to wait for their order for an extended time period, while the other customers in line get their orders much faster [3], [2]. Therefore, the solution must both be efficient in completing whole shopping carts, as well as maintain a reasonably linear order fulfilment between customers.

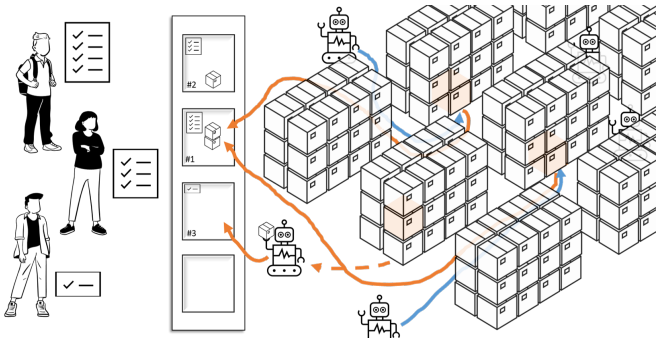


Fig. 1
A conceptual on-demand shopping store where several customers are waiting for their order baskets to be collected. Bob (top) placed his order first (#1) and is being prioritized over Alice (middle) who placed her order (#2) after Bob. Since neither of these have waited long, Mallory (bottom) is also prioritized to get his single item picked. The multi-agent system balance the ordering of basket orders and the time it takes to complete a full basket, as to maximize customer satisfaction through fairness and timely basket picking.

The problem is similar to the well-studied automated warehouse problem [4], where an online order is sent to an Internet-based retail store. The order is then picked by an automated multi-agent logistic system consisting of a large number of robots which move between standardized storage units of commodities without colliding with each other. Upon completion of an order, it is dispatched by for example mail when a courier shows up to collect the current days shipments. The problem is known in the literature as the *Multi-Agent Pickup and Delivery* (MAPD) problem [5], as an extension to the well known Multi-Agent Path Planning problem. To solve this problem, a wealth of approximate methods has been devised, and many of those methods have been deployed in real-world warehouse systems [6]. However, none of the methods in the literature considers the basket-membership of tasks. Rather, the methods and

[1]Department of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden
[2]NearbyStore Sverige AB, Linköping, Sweden
*Corresponding author: mattias.tiger@liu.se

the evaluation metrics treats each task (collecting one item) in isolation. Indeed, this may be reasonable from the perspective of an automated warehouse for online ordering, where it is sufficient that the online order (the basket) is handled to completion in time for package pickup, which may happen only once or twice per day. For a customer waiting for the basket to be picked in a nearby store, however, the latency between order placement and basket completion must be low. The order of basket completion must also be seen as fair, since the customer will observe other customers as they get their baskets delivered. To this end, we propose an improved formalization of the MAPD taking into account baskets, called Basket-MAPD, and formalize the notion of joint *basket completion time* and *unfairness of a waiting customer*, both of which we want to minimize. We further study state-of-the-art methods for MAPD in the context of this novel problem formulation, and propose an improved method that significantly outperforms state-of-the-art. To improve evaluation methodology for the community, we further propose a new benchmark with added realism of lifelong Basket-MAPD derived from real data sources. The proposed method is demonstrated to achieve real-time performance (taking less than 1 second to update given a new added order) on realistic scenarios which are based on real data of customer order patterns and the setup of a real-world on-demand automated grocery store.

The main contributions of this work are: 1) the novel problem of basket-aware MAPD with a consumer satisfaction metric for on-demand automatic shopping stores is proposed and formalized; 2) a realistic and through benchmark to evaluate methods with respect to this problem in various conditions constructed by simulation made realistic using real-world data; 3) a comparative evaluation of three state-of-the-art methods (MCA, PIBT, MLA*) on this benchmark; and 4) a novel method B-PIBT, extending PIBT, which significantly outperforms the state-of-the-art for this problem.

II. Related Work

Advances in automated multi-agent logistics were a key component to making automated warehouses possible [6]. Over the last decades they have changed from being a novelty [4] to being commonplace in the online retail business [7]. These automated warehouses utilize different approaches to solving the problems regarding the path and task planning as well as route execution for multiple robots. Some of the problems that arise in this field are related to collision and deadlock avoidance. In these places, humans are typically prohibited from entering.

Order-picking and delivery in a warehouse setting involves simultaneously solving the *task assignment* problem [8] and the *multi-agent pathfinding* (MAPF) problem [9]. Furthermore, new orders to pick will arrive during execution of previously received orders. The problem is consequently an online multi-agent problem. Online MAPF [10] and lifelong MAPF [5] are well studied problems. In the former, all tasks (navigation goals) are fixed (or assigned) but new agents can appear or disappear. In the latter, new tasks are received incrementally and task allocation between the agents become a challenging problem.

A common view of the automated warehouse problem is as an lifelong version of the MAPF problem, where new tasks to be picked upped and delivered are added during the run-time of the multi-agent system. This lifelong extension, the *Multi-Agent Pickup and Delivery* (MAPD) problem [5], has been actively studied [11], [6], [12], [13] in the literature. It is a combination of the *task allocation problem* [8] and the *vehicle routing problem* [14]. Among the most recent advancements of search-based MAPD solvers are *MCA* [15], heuristics-based task assignment with online improvement, multi-label A* (MLA*) [16], and priority inheritance with backtracking (PIBT) [17].

In this work we limit the scope to robots that have a capacity of carrying a single item (single-task). Work on multi-item MAPD [18] and the capacity-constrained vehicle routing [12] are extensions that considers robot capacity. Another possible extension we do not consider is appearance and disappearance of robots during online MAPF [10].

III. Basket-Aware MAPD

Informally, a *Basket Multi-Agent Pickup and Delivery* (Basket-MAPD) problem instance is a MAPD problem instance in which tasks are annotated with basket membership, i.e. they are part of the same purchase order. Note that we use the term 'basket' rather than 'batch' because the tasks are not necessarily executed in batches. A solution to a Basket-MAPD problem is a plan that results in the delivery of all tasks, which is similar to solutions for ordinary MAPD problems. The distinction lies in the solution evaluation metric. Here, we use the novel *basket ordering weighted error* (BOWE), which takes into account the time it takes to complete a basket and how timely that basket is completed relative to other baskets, representing fairness. This paper seeks to adapt and compare MAPD algorithms to different families of Basket-MAPD problem instances, where the goal is to minimize the BOWE scores of the solutions returned by those algorithms.

A. Formalization of the Basket-MAPD Problem

A *Basket-MAPD problem instance* $\Gamma = (G, A, \mathcal{T})$ can be formalized as a set of agents $A = \{a_1, a_2, \ldots, a_n\}$ in an environment represented by an undirected reflexive graph $G = (V, E)$, consisting of locations V that can be traversed via edges E. We let $V^{\perp} \subset V$ denote the non-empty set of *pick-up locations* and $V^{\top} \subset V$ the non-empty set of *delivery locations*, where additionally $V^{\perp} \cap V^{\top} = \varnothing$. A (time-invariant) *task* $\tau_j \in \mathcal{T}$ is a tuple (t_r, l_p, l_d, b) consisting of a task release time $t_r \geq 0$, a pick-up location $l_p \in V^{\perp}$, a delivery location $l_d \in V^{\top}$, and a incrementing basket identifier $b \in \mathbb{N}$. We use the corresponding short-hand funtions $t_r(\tau)$, $l_p(\tau)$, $l_d(\tau)$, and $b(\tau)$ for any task τ. Two tasks $\tau_i, \tau_j \in \mathcal{T}$ belonging to the same basket are required to have the same release time, i.e. $b(i) = b(j) \rightarrow t_r(i) = t_r(j)$. We can then also define an aggregation function $\beta(\mathcal{T}, b') = \{\tau \in \mathcal{T} \mid b(\tau) = b'\}$, which allows us to group all tasks

contained within any given basket identified by b'. Note that an ordinary MAPD problem instance can be extended to a Basket-MAPD problem instance by setting $b(\tau_i) = i$ for every task $\tau_i \in \mathcal{T}$.

Let $\Gamma^{\mathcal{I}}$ denote the set of valid solutions for a problem instance Γ. A valid solution γ to a Basket-MAPD problem Γ (i.e. $\gamma \in \Gamma^{\mathcal{I}}$), is modeled as a tuple $(\Gamma, \mathcal{V}, \mathcal{L}, T)$, where $\mathcal{V} = \{\rho_1, \rho_2, \dots, \rho_n\}$ is the set of agent *position histories*, $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_n\}$ is the set of agent *load histories*, and $T \subset \mathbb{N}$ represents a *timeline* as a sequence of discrete time-points. Here $\rho_i = (v_1, v_2, \dots)$ represents a path as a sequence of edge traversals in $(v_1, v_2) \in E$ by agent a_i. The location of an agent a_i at a given time-point is formalized as a mapping $l_i : T \mapsto V$ from time-points to locations for each agent $a_i \in A$. Agents are allowed to traverse one edge each time-point, and may not collide, either by ending up in the same location or by traversing the same edge, e.g. to swap places;

$$l_i(t) \neq l_j(t),$$
$$l_i(t+1) = l_j(t) \rightarrow l_i(t) \neq l_j(t+1),$$

for all agents a_i, a_j and time-points $t \in T$. Each agent a_i furthermore has a carrying capacity, denoted by $cap(a_i) > 0$, specifying the number of tasks agent a_i is able to carry at any given time-point. Recall that we limit our scope to robots that have a carrying capacity of one, i.e. $cap(a_i) = 1$ for all agents a_i. The load histories $\ell_i = (l_1, l_2, \dots)$ represent the tasks $l_k \subseteq \mathcal{T}$ that an agent a_i is carrying, where $|l_k| \leq cap(a_i)$ for all values of k. An agent is allowed to pick up a task τ from a pickup location $l_p(\tau)$ on or after that task's release time $t_r(\tau)$ if its carrying capacity allows it, and the agent will keep carrying the task until dropping it off at a drop-off location $l_d(\tau)$, where the time-point of the drop-off is denoted by $t_d(\tau)$. A task that is held by one agent cannot be held by another agent. A solution is valid if it adheres to all of the aforementioned constraints and all tasks in $\mathcal{T}$ have a defined drop-off time.

B. Basket Ordering Weighted Error (BOWE)

We can use various quality metrics to compare valid solutions. In this paper, we consider the *basket ordering weighted error* (BOWE) metric, which combines *basket service time* (BST) measuring boredom, i.e. how long it takes to complete a basket); and *basket linearity error* (BLE) measuring fairness, i.e. how timely the basket is completed relative to other baskets. Let $\gamma \in \Gamma^{\mathcal{I}}$ denote a valid Basket-MAPD solution and let b denote a basket identifier. BST requires knowledge of when the tasks belonging to a basket of interest were delivered, and can thus be formalized as

$$\mathsf{BST}(\gamma, b) =_{def} \max_{\tau \in \beta(\mathcal{T}, b)} \left(t_d(\tau) - t_r(\tau)\right). \quad (1)$$

Similarly, BLE requires knowledge of when tasks within a basket were delivered relative to when they were released, but additionally depends on ranked orderings for basket finishing times and basket release times, i.e. the time-point at which the final task in a basket was completed and the time-point at which the first task in a basket was released respectively. Let $rank_f$ and $rank_r$ represent the rankings of basket finishing times and basket release times respectively; the first basket b to be finished receives a ranking $rank_f(\gamma, b) = 1$, the second basket b'; $rank_f(\gamma, b') = 2$, and so on, with the same holding for $rank_r$-scores. Then the BLE score for a basket b is defined as

$$\mathsf{BLE}(\gamma, b) =_{def} \max(0, rank_f(\gamma, b) - rank_r(\gamma, b)). \quad (2)$$

If all baskets are finished in the same order they are released, the BLE will be 0 for every basket.

Averaging over linear combinations of different cost metrics is common in the literature. However, combining ranking and non-ranking matrices is nontrivial, and uncommon. A suitable weighting parameter between BST and BLE pose a potentially difficult design choice. In this work we opt for weighting BLE *with* BST, capturing the intuition that being more out-of-order is substantially worse if the completion time is also larger. Consequently, we define BOWE as

$$\mathsf{BOWE}(\gamma, b) =_{def} (\mathsf{BLE}(\gamma, b) + 1)\mathsf{BST}(\gamma, b) \quad (3)$$

An optimal solution to a Basket-MAPD problem instance is one which minimizes its basket-averaged BOWE score:

$$\arg\min_{\gamma \in \Gamma^{\mathcal{I}}} \left(\frac{\sum_{b=1}^{B} \mathsf{BOWE}(\gamma, b)}{B} \right), \quad (4)$$

where $B = \max_{\tau \in \mathcal{T}} b(\tau)$.

IV. Solvers

A *Basket-MAPD Solver* is an algorithm that takes a Basket-MAPD problem Γ and produces a solution $\gamma \in \Gamma^{\mathcal{I}}$ that can then be evaluated using evaluation metrics such as BOWE. There already exist a number of MAPD solvers, but recall that these do not take into account baskets. We therefore first consider a pre-existing approach to handling 'singleton' baskets, i.e. baskets containing a single item, before extending it to larger basket sizes.

A. Priority Inheritance with Backtracking (PIBT)

The *Priority Inheritance with Backtracking* (PIBT) [17] algorithm (shown in Algorithm 1) is an iterative algorithm that solves MAPF and MAPD using priority inheritance and backtracking. PIBT starts by updating each agent's priority. The priority is increased with one if the agent has not arrived at its destination yet, else it is reset to its starting value, the tie-breaker ϵ. The next step is to determine the agents' next location, which is done sequentially in decreasing order from highest priority agents to lowest priority agents. When doing this, agents with lower priority try to avoid nodes that agents with higher priority have reserved. This updating of priorities and calculation of next locations is called the top-level procedure and is repeated every timestep.

PIBT is guaranteed [17] to deliver each task an agent is assigned to its destination within a finite number of time steps, if it is possible to do so. Collisions are solved using the backtracking procedure of the algorithm.

Algorithm 1 PIBT

Require: Problem instance Γ with agent goals $\{g_1, \ldots, g_n\}$
For each time-point τ; repeat until terminates
1: Update priorities for all agents A and sort accordingly
2: **while** there are agents that has not planned a path **do**
3: $a_i \leftarrow$ highest priority free agent in A
4: PIBT$(a_i, \perp)$
5: **end while**

6: **procedure** PIBT(a_i, a_j)
7: $C \leftarrow$ neighboring vertices for $l_i(\tau)$
8: sort C in increasing order of $d(u, g_i)$ where $u \in C$
9: **for** $v \in C$ **do**
10: Avoid vertex conflict or **continue**
11: Avoid swap conflict with a_j or **continue**
12: $l_i(\tau + 1) \leftarrow v$
13: **if** $\exists a_k$ s.t. $l_k(\tau) = v \wedge l_k(\tau + 1) = \perp$ **then**
14: **if** PIBT(a_k, a_j) is invalid **then**
15: **continue**
16: **end if**
17: **end if**
18: **return** valid
19: **end for**
20: $l_i(\tau + 1) \leftarrow l_i(\tau)$
21: **return** invalid
22: **end procedure**

B. Basket-PIBT (B-PIBT)

Because PIBT is designed for MAPD problems, it is unware of tasks' basket-membership and consequently handles each task in isolation. The selection of the highest priority agent-task pair is done at line 3 in Algorithm 1. Basket-awareness is introduced by replacing line 3 with Algorithm 2 where basket prioritization is used instead of basket-unaware task prioritization:

$$\frac{(1 - K) \cdot \mathsf{BOWE}}{K \cdot \mathsf{dist} + 1}, \tag{5}$$

where BOWE is the BOWE-value for the potential agent–task-allocation, and K is a weighting parameter. If $K = 0$ then only BOWE matters: if $K = 1$ then only the distance between the agent and the current task matters. The parameter K may be tuned for a given map using Bayesian optimization over the search space $K \in [0; 1]$. Bayesian optimization [19] is a gradient-free optimization method powered by probabilistic inference, where the uncertainty of the objective function is modelled across the input range. HyperOpt [20] is a widely used parameter optimization library implementing Bayesian optimization. It is most often used for parameter tuning in machine learning, but the applicability of Bayesian optimization is widespread [19], [21] and in this work it is used as an automatic method to adapt the MAPD algorithm to the specific domain. One major advantage of using Bayesian optimization is that there is no longer a need for the *ad-hoc* hand-tuning of parameters.

Algorithm 2 Task Assignment of B-PIBT

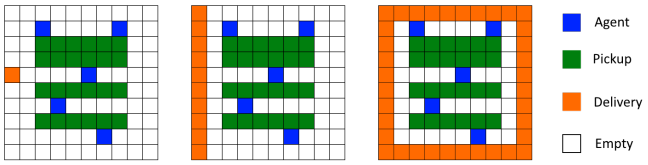
Require: Unassigned tasks T_u, Agent set A
1: $H \leftarrow$ Assignment heaps per agent
2: Initialize each agent heap H_a with $|T_u|$
3: **for** every pair of agent and unassigned task **do**
4: $h_a \leftarrow$ (agent, task) assignment
5: Insert h_a in H_a using basket prioritization Eq. (5)
6: **end for**
7: **while** there are unassigned tasks left **do**
8: Assign best h_a assignment (agent, task) $\forall H_a \in H$
9: Remove h_a from the heap H_a
10: **end while**

V. Evaluation

To evaluate the effectivenes of B-PIBT, we compare it against other MAPD solvers using various metrics including BOWE. We also vary the Basket-MAPD problem instances to be solved by restricting the values for G, A, and $\mathcal{T}$.

A. Measuring Floorplan Configuration Impact

For the first set of evaluations we consider the impact of restricting G to three classes: 1) single-dropoff, where there exists a single dropoff location in $V^{\top}$; 2) column-dropoff, where the dropoff locations form a line; and 3) frame-dropoff, where the dropoff locations follow the entire perimeter of the environment. Instances of the three different classes are illustrated in Figure 2.



(a) Single (b) Column (c) Frame

Fig. 2
The three types of delivery points. The orange squares are delivery points, the green are pickup locations, the blue are agents and the white ones are empty locations

To test the impact of various floorplan configurations, we first fix G to the shown 11x11 grid-shaped floorplans in accordance with the Single, Column, and Frame classes. We then fix the number of tasks to be $|\mathcal{T}| = 50$, where each task belongs to a unique basket, making the problem equivalent to an ordinary MAPD problem. Basket release times are also assumed to follow a release frequency of 1; meaning a basket is released at every time-point. The number of agents is varied such that $|A| \in \{1, 3, 5, 10, 20\}$. This yields a set of Basket-MAPD problem instances $\{\Gamma_1, \Gamma_2, \ldots\}$; one for each permutation, which can further be broken down into $\mathbf{\Gamma}_{\mathsf{single}} \cup \mathbf{\Gamma}_{\mathsf{column}} \cup \mathbf{\Gamma}_{\mathsf{frame}}$ for the three classes of floorplans described by G.

Table I shows the evaluation of the results produced by the MCA, MLA*, and PIBT algorithms grouped by problem

TABLE I
Results showing Service Time (ST), Makespan (MS) and runtime for the three floorplan classes $\mathbf{\Gamma}_{\mathsf{single}}$, $\mathbf{\Gamma}_{\mathsf{column}}$, and $\mathbf{\Gamma}_{\mathsf{frame}}$, using a 11x11 map with frequency of 1, 50 tasks total, aggregated over the number of agents in $\{1, 3, 5, 10, 20\}$. The †-symbol indicates the best values across floorplan classes.

		ST	MS	runtime
$\mathbf{\Gamma}_{\mathsf{single}}$	MCA	957	2897	537533
	MLA*	3185	8432	467455
	PIBT	**935**	**2836**	**46**
$\mathbf{\Gamma}_{\mathsf{column}}$	MCA	**805**	**2558**	533365
	MLA*	860	2771	7837
	PIBT	839	2602	**37**
$\mathbf{\Gamma}_{\mathsf{frame}}$	MCA	†**624**	2077	515292
	MLA*	721	2064	352
	PIBT	720	†**2061**	†**26**

sets $\mathbf{\Gamma}_{\mathsf{single}}$, $\mathbf{\Gamma}_{\mathsf{column}}$, and $\mathbf{\Gamma}_{\mathsf{frame}}$. The metrics shown for each algorithm are averaged over five runs and include Service Time (ST), Makespan (MS), and runtime (in seconds). The Service Time is calculated similar to BST but for individual tasks:

$$\mathsf{ST}(\gamma) =_{def} \sum_{\tau \in \mathcal{T}} \left(t_f(\tau) - t_r(\tau)\right). \tag{6}$$

The Makespan metric is calculated to be the number of timesteps between the first released task to the last finished task:

$$\mathsf{MS}(\gamma) =_{def} \max_{\tau \in \mathcal{T}}(t_f(\tau)) - \min_{\tau \in \mathcal{T}}(t_r(\tau)). \tag{7}$$

Note that while MCA and MLA* perform poorly runtime-wise, the latter improves significantly in the Frame class of floorplans compared to Column and Single. PIBT has far lower runtimes and performs comparably to MCA and MLA* in the ST and MS metrics.

B. Simulating Realistic Baskets

To evaluate the performance of the different solvers on realistic data, we need to simulate varying basket sizes and times between basket releases in a way that follows real-world behavioral patterns. For this reason, we made use of two datasets: 1) the *Instacart Market Basket Analysis* [22]; and 2) the average number of daily orders of a prototype automated grocery store. Instacart's dataset contains data on over 3 million grocery store orders from from more than 200,000 customers. Data pertaining to groceries (rather than clothing, for example) was used to model baskets. Figure 3 shows a histogram of order sizes (i.e. the number of tasks per basket) and the frequency of orders (i.e. basket releases) across a 24-hour period, and consequently models temporal behavioral patterns of customers.

In order to simulate realistic baskets, we first need to approximate the behavioral data indicating when customers place orders and how large those orders are. Each arriving order contains a number of products that are represented as baskets and tasks respectively. The number of products (i.e. tasks) sold per order (i.e. basket) vary throughout the day. The basket size is modeled using a negative binomial distribution, denoted $\mathbf{NB}(r, p)$, with different parameters for

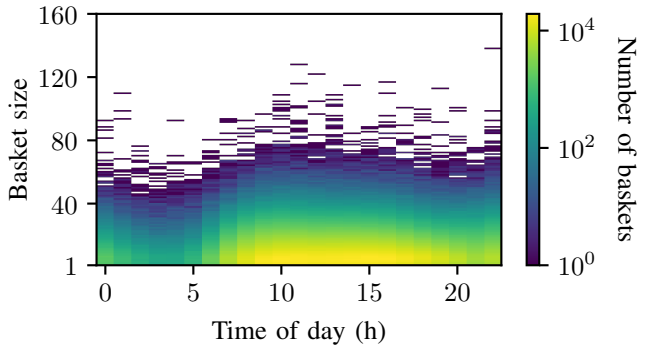


Fig. 3
One-hour histograms of basket (order) counts and sizes compared to time of day based on Instacart's dataset.

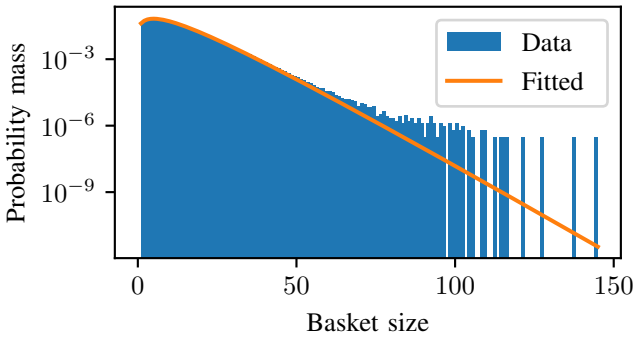


Fig. 4
Fitted probability mass function of the aggregate of all time points in the Instacart dataset, showing the (time-invariant) probability of drawing a particular basket size.

each one-hour period of the simulation, informed by the observed behavioral patterns in the dataset. The parameters are estimated using maximum-likelihood:

$$p_h = \frac{\overline{X_h}}{\mathbf{Var}(X_h)}, \tag{8}$$

$$r_h = \frac{p_h \overline{X_h}}{(1 - p_h)}, \tag{9}$$

where X_h is a set of observed order sizes during the hour h. Since the support of the negative binomial distribution contains 0, which is not a valid basket size, 1 is subtracted from the basket sizes when estimating the parameters, and 1 is added during sampling. An example distribution showing the probability of a particular basket size is shown in Figure 4 for a specific one-hour period during the day, where the distribution has been fit against the data for a specific hour.

While the above model contains information on an hourly basis, the available data needed to determine the exact timing of when orders arrive to the system—corresponding to basket release times—is unfortunately scarce. Here an assumption is therefore made that orders arrive in bursts rather than uniformly throughout any given hour. The number of orders arriving at a time interval N_t is therefore modeled using a

TABLE II
The $\mathbf{\Gamma}_{\text{proto}}$-benchmark configuration space representing scenarios in the prototype automated grocery store.

Symbol	Description	Distribution
$\lvert V\rvert$	Map size	$\{11 \times 37\}$
$\lvert V^{\perp}\rvert$	Number of pickups	$\{165\}$
$\lvert A\rvert$	Number of agents	$\{5, 10, 15, 20\}$
$\lvert \mathcal{T}\rvert$	Number of tasks	$\{500\}$
λ_h	Frequencies	$\{0.2, 0.5, 1, 2, 5\}$
B	Basket sizes	$\{2, 6, 10, 14, 18\}$

Poisson distribution, again with different parameters for each hour of the simulation:

$$N_t \sim \mathbf{Poisson}(\lambda_h), \tag{10}$$

where $\lambda_h = \overline{N} p_h$; $p_h = N_h / c \sum_{i=8}^{20} N_i$; and c denotes the (configurable) number of intervals in an hour. This assumes that the orders' arrivals are independent, that the rate is constant and that the rate itself is independent from any order arriving. In our simulation each interval lasts for a second.

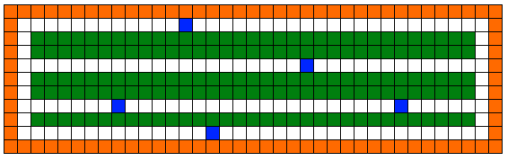


Fig. 5
The real-world prototype automated grocery store's 11×37 floorplan with 5 agents and 170 pickup locations.

We now have models that allow us to simulate realistic order times and sizes across a 24-hour period. These models have been used to generate a collection of Basket-MAPD problem instances $\mathbf{\Gamma}_{\text{proto}}$ that have been varied in accordance with Table II, and where the floorplan is modeled after the prototype automated grocery store, shown in Figure 5. For these problem instances, the map size is assumed to remain constant as it follows the floormap of the prototype automated grocery store. Applying Bayesian optimization to BOWE's weighting parameter yields $K = 6.6 \times 10^{-3}$ for this map, which shows that BOWE heavily influences the allocation without completely excluding distance information.

As with the previous set of experiments for the Basket-MAPD problem instances, the solvers MCA, MLA*, PIBT, as well as B-PIBT are applied to the Basket-MAPD problem instances in $\mathbf{\Gamma}_{\text{proto}}$, and the results are evaluated against ST, MS, runtime, BST, and BOWE. For the Off-line simulations, the calculations are performed sequentially for each agent. The Lifelong simulations are long-running on-line scenarios that cover a subset of the permutations defined in Table II. The number of agents $\lvert A\rvert$ is fixed to 20 agents, and the map is the same 11×37 map as before, but the scenario durations are increased to 12 hours of 1-second time-steps, resulting in $\lvert T\rvert = 43,200$. The simulation thus models the timing and size of orders arriving to the store during a 12-hour interval (08:00–20:00) during which the simulated store is open. The results are shown in Table III, where the algorithm suffixes '–600' and '–5000' refer to problem instances with 600 and 5000 orders (i.e. baskets) per day.

TABLE III
Results showing Service Time (ST), Makespan (MS), runtime (sec), Basket Service Time (BST), and Basket Ordering Weighted Error (BOWE) for the prototype automated grocery store MAPD problem instances $\mathbf{\Gamma}_{\text{proto}}$.

Off-line $\mathbf{\Gamma}_{\text{proto}}$	ST	MS	runtime	BST	BOWE
MCA	**264**	1593	116704	**604**	5441
MLA*	326	1602	781	646	5645
PIBT	334	**1589**	**175**	691	8417
B-PIBT	565	1898	806	649	**1125**
Lifelong $\mathbf{\Gamma}_{\text{proto}}$					
PIBT-600	**26**	**46711**	**3492**	**43**	**47**
B-PIBT-600	73	46723	4652	91	95
PIBT-5000	**6273**	**57430**	**162916**	22293	12519080
B-PIBT-5000	12247	66349	6277099	**12645**	**36139**

The results show that B-PIBT performs well on the BOWE metric in the off-line case. For the on-line cases, PIBT performs better than B-PIBT for lower daily basket counts, covering more quiet days at the grocery store. However, for busier days, B-PIBT heavily outperforms PIBT, which does not appear to scale very well.

VI. Conclusions

While there exist many solvers for MAPD problems, none consider the basket-membership of individual tasks. Basket-membership awareness is an important property when considering automated shopping stores, in which agents are tasked with picking the items in customer orders while keeping customer satisfaction high.

This paper therefore presented the novel problem of Basket-MAPD, along with a consumer satisfaction metric BOWE based on waiting times and perceived fairness. To find solutions to Basket-MAPD problem instances that minimize BOWE, an algorithm based on PIBT called Basket-PIBT (B-PIBT) was presented and compared against other state-of-the-art solvers. The Basket-MAPD problem instances were also designed to follow real-world behavioral patterns associated with order placements, and considered various shop floorplans in order to study their impacts. We showed that B-PIBT significantly outperforms state-of-the-art alternatives when considering BOWE, while not performing significantly worse in other (basket-unaware) metrics. Since the evaluations were performed in simulations, future work could consider field robotic experiments.

Acknowledgment

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by Grants from the National Graduate School in Computer Science (CUGS), Sweden, the Excellence Center at Linköping-Lund for Information Technology (ELLIIT), the TAILOR Project funded by EU Horizon 2020 research and innovation programme GA No 952215, and Knut and Alice Wallenberg Foundation under Grant KAW 2019.0350.

REFERENCES

[1] N. M. Puccinelli, R. C. Goodstein, D. Grewal, R. Price, P. Raghubir, and D. Stewart, "Customer experience management in retailing: Understanding the buying process," *Journal of Retailing*, vol. 85, no. 1, pp. 15–30, 2009.

[2] S. Stevenson, "What you hate most about waiting in line," Jun 2012. [Online]. Available: https://slate.com/business/2012/06/queueing-theory-what-people-hate-most-about-waiting-in-line.html

[3] E. N. Weiss and C. Tucker, "Queue management: Elimination, expectation, and enhancement," *Business Horizons*, vol. 61, no. 5, pp. 671–678, 2018.

[4] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–19, 2008.

[5] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent System (AAMAS)*, 2017.

[6] P. O. Dusadeerungsikul, X. He, M. Sreeram, and S. Y. Nof, "Multi-agent system optimisation in factories of the future: Cyber collaborative warehouse study," *International Journal of Production Research*, vol. 60, no. 20, pp. 6072–6086, 2022.

[7] Y. Liu, D. Han, L. Wang, and C.-Z. Xu, "HGHA: Task allocation and path planning for warehouse agents," *Assembly Automation*, vol. 41, no. 2, pp. 165–173, 2021.

[8] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, vol. 2, no. 1-2, pp. 83–97, 1955.

[9] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the Twelfth International Symposium on Combinatorial Search (SoCS)*, 2019.

[10] J. Švancara, M. Vlk, R. Stern, D. Atzmon, and R. Barták, "Online multi-agent pathfinding," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, no. 01, 2019, pp. 7732–7739.

[11] O. Salzman and R. Stern, "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 (AAMAS)*, 2020, p. 1711–1715.

[12] A. Farinelli, A. Contini, and D. Zorzi, "Decentralized task assignment for multi-item pickup and delivery in logistic scenarios," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2020, pp. 1843–1845.

[13] H. Ma, W. Hönig, T. S. Kumar, N. Ayanian, and S. Koenig, "Lifelong path planning with kinematic constraints for multi-agent pickup and delivery," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019, pp. 7651–7658.

[14] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[15] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 3, pp. 5816–5823, 2021.

[16] F. Grenouilleau, W.-J. van Hoeve, and J. N. Hooker, "A multi-label A* algorithm for multi-agent pathfinding," in *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 29, 2019, pp. 181–185.

[17] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, 2022.

[18] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5022–5027.

[19] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[20] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*. PMLR, 2013, pp. 115–123.

[21] D. Engelsons, M. Tiger, and F. Heintz, "Coverage path planning in large-scale multi-floor urban environments with applications to autonomous road sweeping," in *Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3328–3334.

[22] J. Stanley, M. Risdal, S. Rao, and W. Cukierski, "Instacart market basket analysis," 2017. [Online]. Available: https://kaggle.com/competitions/instacart-market-basket-analysis