Autonomous Intelligent Systems

ORIGINAL ARTICLE **Open Access**

# Task scheduling for transport and pick robots in logistics: a comparative study on constructive heuristics

Hanfu Wang[1,2,3] and Weidong Chen[1,2,3]*

**Abstract**

We study the Transport and Pick Robots Task Scheduling (TPS) problem, in which two teams of specialized robots, transport robots and pick robots, collaborate to execute multi-station order fulfillment tasks in logistic environments. The objective is to plan a collective time-extended task schedule with the minimization of makespan. However, for this recently formulated problem, it is still unclear how to obtain satisfying results efficiently. In this research, we design several constructive heuristics to solve this problem based on the introduced sequence models. Theoretically, we give time complexity analysis or feasibility guarantees of these heuristics; empirically, we evaluate the makespan performance criteria and computation time on designed dataset. Computational results demonstrate that coupled append heuristic works better for the most cases within reasonable computation time. Coupled heuristics work better than decoupled heuristics prominently on instances with relative few pick robot numbers and large work zones. The law of diminishing marginal utility is also observed concerning the overall system performance and different transport-pick robot numbers.

**Keywords:** Multi-robot task allocation, Multi-robot system, Complex-schedule constraints, Heterogeneous robotic order fulfillment system

## 1 Introduction

In the *Heterogeneous Robotic Order Fulfillment System (HROFS)*, two types of robots with specialized and complementary capabilities exist: 1) *transport robots* with object transfer and transient storage capability, typically automated guided vehicles (AGVs) or autonomous mobile robots (AMRs), that autonomously receive order fulfillment tasks, travel across the workspace and retrieve items from different storage stations and 2) *pick robots* with mobile manipulation capability, typically mobile manipulators, mobile dual-arm robots, hybrid leg-wheel robots or even human workers enhanced by mobile platforms (only for tricky products that is still challenging for robots), that fetch items from storage stations and place them into transport robots' containers. As is verified in our previous research [1], by combining two types of robots, practitioners can achieve the same or better system performance with relatively lower cost.

Targeting the task allocation and scheduling aspect of the HROFS, we formulate the *Transport and Pick Robots Task Scheduling (TPS)* problem, which seeks the collective time-extended task schedule for multiple transport robots and multiple pick robots collaboratively performing order fulfillment tasks. In the TPS problem, as customer orders are characterized by multiple lines of miscellaneous items, both two robot types have to travel to different locations to collect items or perform pick&place operations. Furthermore, as we split object transfer and pick&place operations and assign them to two robot types, to achieve

*Correspondence: wdchen@sjtu.edu.cn
[1]Institute of Medical Robotics, Shanghai Jiao Tong University, Shanghai 200240, China
[2]Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
Full list of author information is available at the end of the article

better performance, it is paramount to enable them to act tightly and synergistically.

According to the multi-robot task allocation (MRTA) taxonomy *iTax* [2], the TPS problem falls in the complex-schedule [CD] category, with single-task robots [ST], multiple-robot tasks [MR], and the time-extended allocation [TA] problem (CD[ST-MR-TA]). *iTax* by Korsah, et al. [2] is the first taxonomy for multi-robot task allocation that explicitly takes issues of inter-task constraints and inter-robot utilities into consideration. Nunes, et al. [3] further propose a novel taxonomy for MRTA problems with temporal and ordering constraints (MRTA/TOC), which belongs to the XD[MT-SR-TA] category in *iTax*. Typical MRTA methods include centralized methods (exact [4], approximate [5], heuristics [6] and metaheuristics [7]), decentralized methods (game-theoretic [8], market and negotiation-based [9]), swarm-based methods [10] and hybrid methods [11, 12]. For research on MRTA with complex-schedule constraints, materials are relatively rare. Jones, et al. [13] solve the time-extended multi-robot coordination with intra-path precedence constraints in the disaster response domain. The robot team comprises of multiple unmanned fire trucks and multiple debris cleaning bulldozers. They propose two methods, a hybrid method incorporating tiered auctions and two heuristic techniques, clustering and opportunistic path planning. The other method is genetic algorithm. Strengths and weakness of both methods are analyzed and verified by simulations.

The TPS problem is also an integrated routing and scheduling problem, and several combinatorial problems are related to it. Vehicle routing problem (VRP) [14] and its variants capture the routing components of the TPS problem. Open shop scheduling problem (OSP) [15] and its variants capture the scheduling components of the TPS problem. The open shop scheduling problem also has many integrated routing and scheduling variants, such as routing open shop scheduling [16], open shop scheduling with sequence-dependent setup times [17] and open shop scheduling with transportation/travel times [18]. These problems can be solved by exact methods [19], approximate methods [16, 20, 21], heuristics [22] and metaheuristics [17, 18, 23, 24].

In our previous work [25] and [1], we use decoupled and coupled methods to plan collective time-extended task schedule respectively. Specially in [1], we propose a rank-minimal heuristic based on tripartite matching. We use genetic algorithm as the solver in each iteration. In practice, decision-making on task level should be made on-the-fly, and obtaining satisfying results in a short time is paramount. For such circumstances heuristic methods are natural choices. In fact, as a typical integrated routing and scheduling problem, there exist many potential heuristics yet to be explored for the TPS problem. However, there is a lack of a comprehensive heuristic algorithm design principles and it is still unclear how to choose among multiple methods. In view of this, in this research we concentrate on designing constructive heuristics, conduct a comparative study, and figure out the most promising heuristic.

This paper is organized as follows. In Section 2, we formulate the TPS problem and introduce the sequence models. In Section 3, we design five constructive heuristics together with complexity analysis. In Section 4, we conduct a comparative study on these heuristics and analyze computational results. Finally, in Section 5, we provide the conclusions and point out possible future work.

## 2 Problem definition

### 2.1 Problem formulation

The TPS problem is defined by three sets, a heterogeneous robot set $\mathcal{R}$, a logistic network $\mathcal{G}$ and an order fulfillment task set $\mathcal{T}$. We explicitly consider the environment or the logistic network because pick robots are associated with their dedicated work zones.

*Logistic Network:* The logistic network is represented as a simple, undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The vertex set $\mathcal{V} = \mathcal{V}_P \cup \mathcal{V}_D$ is a union of the pickup station subset $\mathcal{V}_P$ and the delivery station subset $\mathcal{V}_D$, and $\mathcal{V}_P \cup \mathcal{V}_D = \emptyset$. A delivery station $v \in \mathcal{V}_D$ is a vertex where transport robots receive order fulfillment tasks, start from and return to. A pickup station $v \in \mathcal{V}_P$ is a vertex where two types of robots meet so that pick robots perform pick&place operations, and transport robots collect items. The edge set $\mathcal{E}$ consists of connections between stations and each edge $e \in \mathcal{E}$ is assigned to a travel cost $w(e)$.

*Robots:* The heterogeneous robot set $\mathcal{R}$ consists of $m$ transport robots and $n$ pick robots, i.e., $\mathcal{R} = \left\{R_i^T | i = 1, 2, \cdots, m\right\} \cup \left\{R_j^P | j = 1, 2, \cdots, n\right\}$. A transport robot $R_i^T$ starts from a delivery station and undertakes an order fulfillment task from the task set, travels across the whole workspace as subtasks indicates, rendezvous with pick robots to allow them pick&place items, and finally returns to the delivery station after all items are collected. A pick robot $R_j^P$ is dedicated to a work zone represented as a set of pickup stations $\mathcal{Z}_i$, $\mathcal{Z}_i \subset \mathcal{V}_P$. It picks items from storage (racks, shelves) at corresponding pickup stations, meets with transport robots, and places items into transport robots' containers. We do not allow any two work zones overlap, i.e., $\forall 1 \leq p, q \leq n, p \neq q, \mathcal{Z}_p \cap \mathcal{Z}_q = \emptyset$. Meanwhile, all work zones should cover the pickup station subset, i.e., $\cup_{j=1}^{n} \mathcal{Z}_j = \mathcal{V}_P$.

*Tasks:* Let $\mathcal{T} = \{T_1, T_2, \cdots, T_m\}$ denote the set of $m$ order fulfillment tasks. Currently for one-shot scheduling we only consider tasks with the same transport robot number $m$, and we also presume the transport robot $R_i^T$ has been pre-assigned to the task $T_i \in \mathcal{T}$. This is because task allocation problems (assigning tasks to transport robots) can be solved independently from task scheduling. Each task $T_i$ is a set of several pickup subtasks and one delivery subtask, i.e., $T_i = \{t_{i00}\} \cup \{t_{i1k}, k = 0, 1, \cdots, K_{i1}\} \cup \cdots \cup \left\{t_{ijk}, k = 0, 1, \cdots, K_{ij}\right\} \cup \cdots \cup \{t_{ink}, k = 0, 1, \cdots, K_{in}\}$, where $K_{ij} \in \mathbb{N}^+ \cup \{0\}$ is the number of pickup subtasks in the zone $\mathcal{Z}_j$. A pickup subtask $t_{ijk}$ is defined by its pickup station $v_{ijk}$ and a positive pick&place operate time $\tau_{ijk}$, i.e., $t_{ijk} : \left(v_{ijk}, \tau_{ijk}\right), v_{ijk} \in \mathcal{V}_P, \tau_{ijk} \in \mathbb{R}^+$. Similarly the delivery subtask $t_{i00}$ is defined by its delivery station $v_{i00}$ and a positive operate time $\tau_{i00}$, i.e., $t_{i00} : (v_{i00}, \tau_{i00}), v_{i00} \in \mathcal{V}_D, \tau_{i00} \in \mathbb{R}^+$. Pick&place operate time is simply called operate time afterwards. This task definition caters to real-life order-fulfillment tasks in which customer orders may include multiple lines of items, and multiple orders from different customers can also be combined by the order batching process [26].

*Outputs:* We seek all robots' collective time-extended task schedule $\mathcal{TS} = \{TS_i^T\} \cup \{TS_j^P\}, 1 \leq i \leq m, 1 \leq j \leq n$, in which $TS_i^T$ or $TS_j^P = \left\{\left(ST^*_{ijk}, AT^*_{ijk}, PT^*_{ijk}, CT^*_{ijk}\right)\right\}$ $(* = \{T, P\})$. For subtask $t_{ijk}$ belonging to transport robot $R_i^T$'s task, or in pick robot $R_j^P$'s work zones, $ST^*_{ijk}, AT^*_{ijk}, PT^*_{ijk}, CT^*_{ijk}$ are start time, arrive time, pick time and completion time respectively.

**Problem 1** ***Transport and Pick Robots Task Scheduling (TPS) Problem*** *Given a TPS problem instance $\langle \mathcal{R}, \mathcal{G}, \mathcal{T} \rangle$ as described previously, find a collective time-extended task schedule $\mathcal{TS}$, so that all robots collaborate with each other and accomplish all allocated tasks with the minimization of the makespan $C_{max}$. If task $T_i$'s completion time is $C_i$, the makespan is $C_{max} = \max\{C_i\}, i = 1, 2, \cdots, m$.*

A TPS problem instance with 3 transport robots and 4 pick robots is depicted in Fig. 1. In this scenario, 3 transport robots $R_1^T, R_2^T, R_3^T$ and 4 pick robots $R_1^P, R_2^P, R_3^P, R_4^P$ collaborate to execute 3 order fulfillment tasks $T_1, T_2, T_3$. The objective is to find collective task schedule composed of all 7 robots' schedules, with the makespan minimized.

*Assumptions:* This research is subject to several assumptions. We concentrate on one-shot deterministic task scheduling with all necessary information available. Travel time estimation is sufficient enough for robots to travel between different subtask stations with collisions or deadlocks avoided. Operate times are adequate for pick robots to perform pick&place operations. Work zones for all pick robots are equal.

According to taxonomy *iTax* [2], the TPS problem falls in the CD[ST-MR-TA] category. This is because transport robots' and pick robots' task execution orders all are not predetermined; each robot can merely execute one task at the same time; and each subtask requires two types of robots work at the same time. Complex-schedule constraints exist between tasks and the utilities of all robots are interrelated and interdependent. Each robot's schedule is coupled with others and cannot be independently determined. The search space of task execution orders is exponentially large. Two classes of coupledness exist in the TPS problem: coupledness between two types of robots, and coupledness between routing and scheduling components.

## 2.2 Sequence models

To establish the partial relations between two subtasks for all robots unanimously, we can use directed acyclic graph (DAG). As is stated in our previous work [1], the TPS problem and the open shop problem share the same property: two types of orders (transport robot subtask orders and pick robot subtask orders, or job orders and machines orders) should be determined by the task scheduler. For open shop scheduling problems, the permutation list [27] and the rank matrix (can be bijectively mapped to a sequence graph) [20, 22, 23, 28–31] are commonly used models. The permutation list is simple and easy to build, however, it suffers from a notorious drawback that is high redundancy: multiple permutation lists possibly mean the same sequence. Oppositely, the rank matrix is non-redundant, yet to guarantee feasibility and nonredundancy, more pre-processing and post-processing algorithms are needed.

For the TPS problem, as the combinatorial space is much more complicated than open shop scheduling problems, non-redundancy property is more preferable. Using the rank matrix models could provide more insightful descriptions on the solution structure. Meanwhile, as they can always provide feasible and non-redundant solutions beforehand, efficient algorithms can be developed. For the TPS problem, *block sequence graph* and *block rank matrix* are extended from sequence graph and rank matrix models for open shop problems.

A block sequence graph is *feasible* if the corresponding directed acyclic graph obtained by the union of all transport robots' subtask orders and pick robots' subtask orders is acyclic. block sequence graph differs from (preemptive) sequence graph [20] in that direct precedence constraints between two split operations need not be considered, as a result, the search space of block sequence graph is larger.

For each block sequence graph, there exists a block rank matrix capturing its structural properties algebraically. In a block rank matrix, an entry, namely rank $s_{ijk}$ means that in the corresponding block sequence graph a path to subtask $t_{ijk}$ with a maximal number of subtasks includes $s_{ijk}$ subtasks. A block rank matrix can always be transformed into a permutation list by topological sorting or linearization. The block sequence graph and its corresponding

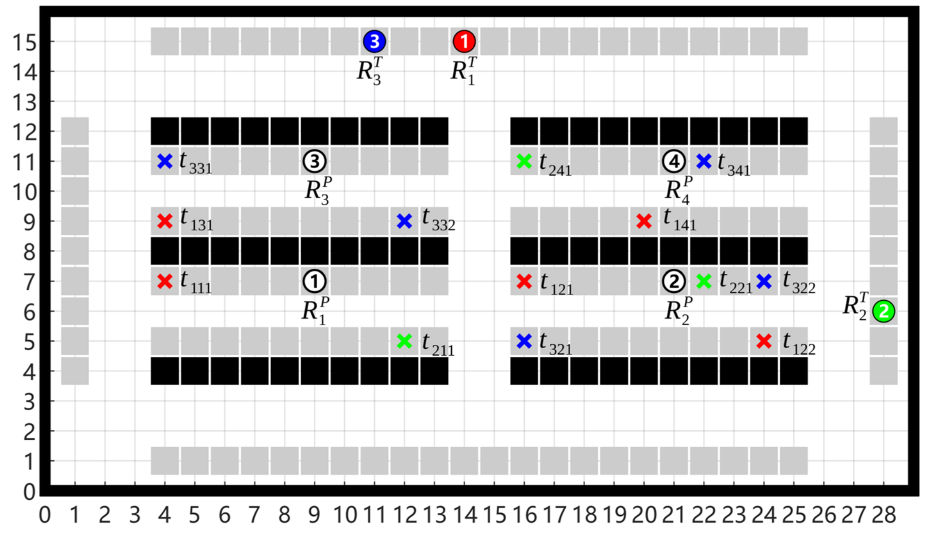

**Fig. 1** A TPS problem instance with 3 transport robots (red, green and blue discs with number 1,2,3), 4 pick robots (white discs with number 1,2,3,4) and 3 tasks (red, green or blue crosses). Transport robots and their corresponding order fulfillment tasks are depicted by the same color (red, green and blue). Pick robots are confined in their corresponding work zones $\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3, \mathcal{Z}_4$, each of which is an aisle between two opposite racks (black bars). Transport robots start from the delivery stations (gray squares in periphery of the workspace), and in contrast, pick robots start from the pickup stations (gray squares near the racks). Each task contains several subtasks possibly in different work zones

block rank matrix are shown in Fig. 2 and Eqs. (1), (2).

$$\boldsymbol{TO} = \begin{bmatrix} \{1\} & \{5,2\} & \{4\} & \{3\} \\ \{1\} & \{3\} & \cdot & \{2\} \\ \cdot & \{3,5\} & \{4,1\} & \{2\} \end{bmatrix},$$

$$\boldsymbol{PO} = \begin{bmatrix} \{1\} & \{5,1\} & \{3\} & \{2\} \\ \{2\} & \{4\} & \cdot & \{3\} \\ \cdot & \{2,3\} & \{2,1\} & \{1\} \end{bmatrix}, \tag{1}$$

$$\boldsymbol{S} = \begin{bmatrix} \{1\} & \{7,2\} & \{5\} & \{3\} \\ \{2\} & \{6\} & \cdot & \{4\} \\ \cdot & \{3,5\} & \{4,1\} & \{2\} \end{bmatrix} = \left[\begin{array}{c|cc|cc|c} 1 & 7 & 2 & 5 & 0 & 3 \\ 2 & 6 & 0 & 0 & 0 & 4 \\ 0 & 3 & 5 & 4 & 1 & 2 \end{array}\right]. \tag{2}$$

## 3 Constructive heuristics

In this section, we design efficient constructive heuristics for the TPS problem utilizing previous sequence models. Based on whether two types of robots are scheduled at the same time, constructive heuristics can be divided into coupled and decoupled heuristics. Typical discrete operations include matching, append and insertion. Some of these operations have been designed for open shop scheduling problems in [22]. For integrated routing and scheduling problems or variants of open shop scheduling [16, 21, 32], although progresses have been made to find approximate methods with bounded suboptimality and polynomial computation time, these performance bounds are too broad, and far from the need of practical robotic deployment. Meantime, to obtain approximate results, some unrealistic assumptions or limitations have to be made. As a result, we conduct a comparative study on constructive heuristics, and find out the most promising heuristic that can produce good solutions in reasonable time.

In different heuristics, coupledness are dealt differently. In decoupled heuristics, two types of robots are separately scheduled. In coupled append or insertion, tasks are scheduled iteratively one by one. As of matching, multiple tasks are scheduled iteratively. Except the coupled insertion heuristic, other heuristics do not change the

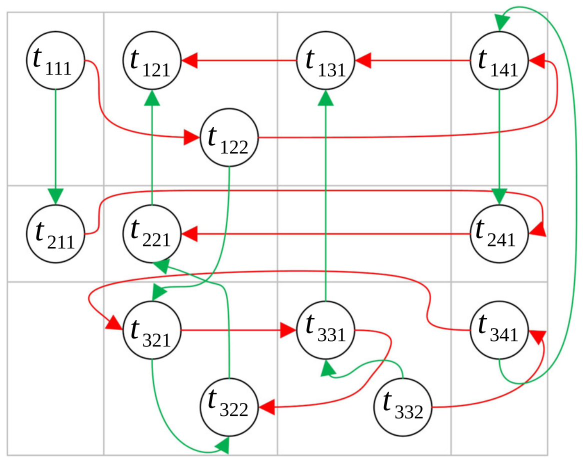

**Fig. 2** Block sequence graph for a feasible task sequence in Example 1. Transport robot subtask orders are represented as horizontal red arcs, and pick robot subtask orders are represented as vertical green arcs. Gray rectangles are blocks. The union of arcs should be acyclic

**Table 1** A summary of constructive heuristics for the TPS problem

| **Heuristics** | **Wether two robot types are scheduled simultaneously** | **Discrete operations** | **Subtasks scheduled in each iteration** | **Whether previous established partial task schedule is changed** |
|---|---|---|---|---|
| Decoupled, Greedy | no | append | one | no |
| Decoupled, Bipartite Matching | no | matching | multiple | no |
| Coupled, Tripartite Matching | yes | matching | multiple | no |
| Coupled, Append | yes | append | one | no |
| Coupled, Insertion | yes | insertion | one | yes |

established partial task schedule. A brief summary of the proposed constructive heuristics are listed in Table 1.

### 3.1 Decoupled, greedy heuristic

For the multi-robot system involving two types of robots, decoupled or hierarchical planning are commonly used in most previous robotics literature [13, 33, 34]. Decoupled methods mean that firstly making decisions for one robot team, and then for the other robot team, with already decided variables acting as constraints. Using this idea, decoupled, greedy heuristic is designed as follows. 1) For each transport robot, find its subtask execution order separately by solving a traveling salesman problem, with the minimization of its routing component. This could be done by using existing traveling salesman problem solvers, either heuristics or metaheuristics. 2) Given all transport robots' subtask orders, pick robots' task scheduling problem can be regarded as a *precedence-constrained task scheduling problem* [35] belonging to the cross-schedule category in *iTax*. Pick robots' subtask orders are constructed iteratively by three rules, without violation of previously established transport robots' subtask orders. (a) *Free subtask first rule.* Only subtasks that have no precedent subtasks in transport robot orders (i.e., "free" subtasks in [35]) are released. (b) *Busy rule.* Pick robots are not allowed to be idle if they have unfinished free subtasks. (c) *Shortest rendezvous time first rule.* When more than one free subtasks are in the work zone, the pick robot always chooses the subtask that requires minimal rendezvous time for the transport-pick robot pair. Free subtasks that have been appended to the pick robot subtask orders are removed form unscheduled subtask set. The *free-select-remove* process continues until all subtasks are scheduled.

*Formal Analysis:* We use an ordinary matrix in Eq. (2) to analyze the time complexity of all algorithms. The dimension of an ordinary matrix transformed from a block rank matrix of Problem 1 is $m \times nK_{max}$, in which $K_{max} = \max_{1 \le j \le n}\left(\max_{1 \le i \le m} K_{ij}\right)$ is the maximum block width. Then in the worst case each transport robot has at most $nK_{max}$ subtasks, and each pick robot has at most $mK_{max}$ subtasks.

**Proposition 1** *The decoupled greedy heuristic terminates in polynomial time.*

*Proof* In step (a), suppose the traveling salesman problem is solved by the famous Christofides heuristic, which is an $O\left((nK_{max})^3\right)$ and 1.5-optimal algorithm. Totally the time complexity in step 1 is $O\left(mn^3K_{max}^3\right)$. To generate a complete schedule, the free-select-remove process is run for $|\mathcal{T}| \le m \times nK_{max}$ iterations. Note that here $|\mathcal{T}|$ denotes the number of subtasks, rather than the number of tasks. In each iteration, at most $m$ subtasks are released for $n$ pick robots. The total time complexity for step (b)(c) is simply at most $O(mnK_{max} \cdot mn)$. Finally, the worst-case time complexity of the decoupled greedy heuristic is $O\left(mn^3K_{max}^3 + m^2n^2K_{max}\right)$. Then this proposition is proved. □

### 3.2 Coupled, tripartite matching heuristic

The matching heuristic could generate rank-minimal task schedules. These rank-minimal schedules are of great interest because they could generate good task schedule in which the number of robots in the longest robot-subtask chain is minimized. The coupled tripartite matching heuristic works as follows. Let $G^* = (\mathcal{R}_{\mathcal{T}} \cup \mathcal{R}_{\mathcal{P}} \cup \mathcal{T}, E^*)$ be the complete tripartite graph with the set of transport robots, pick robots and subtasks as three partitioned vertices. Each edge $e \in E^*$ is assigned with a weight that equals to the operate time of the subtask, plus the travel time between the robot's current vertex and the subtask vertex. In the tripartite matching, we consecutively determine a maximal number of subtasks (i.e., transport-pick robot pair and mutual subtask) having the same rank (increasingly from 1 to $r_{max}$). In each iteration, at most $\max\{m, n\}$ subtasks are determined. This is done by solving the weighted tripartite maximum cardinality matching problem. Using this rule, the rank-minimal schedule is generated, i.e., the greatest value of the rank of the sequence is minimal.

In each iteration, based on how to determine the tripartite matching, different objectives may be used and thus many variants exist. We use the dynamic partial makespan as objective function, as Eq. (3) shows. The main idea of MINDYMAX is to choose robots and subtasks with minimal partial makespan in each iteration with all scheduled and chosen subtasks considered.

$$\begin{aligned}\textsc{MinDyMax}: \min \max_{i,j,k \in M} \left(ha_{ijk} + \tau_{ijk}\right), \\ i = 1, 2, \cdots, m, j = 1, 2, \cdots n, k = 1, 2, \cdots K_{max}\end{aligned} \tag{3}$$

*Formal Analysis:* It is well-known that solving a tripartite matching problem is generally NP-hard, so this heuristic may run infinitely based on the problem structure. However, as we do not need to solve the tripartite matching on complete tripartite graphs, the matching can be solved optimally in reasonable time. Meanwhile, as more and more subtasks are scheduled, the size of the tripartite matching problem decreases. As is shown in the latter empirical study, the coupled tripartite matching could generate task schedules in half minute.

## 3.3 Decoupled, bipartite matching heuristic

The decoupled, bipartite matching heuristic combines the main idea of two previous heuristics: making decisions for two types of robots separately; and obtaining a rank-minimal task schedule by matching. The first step is the same as step (a) in the decoupled, greedy heuristic. Because transport robots' subtask execution orders are pre-determined and do not change during scheduling, in each iteration we solve a bipartite maximum cardinality matching problem and at most $\max\{m, n\}$ subtasks are determined.

*Formal Analysis:* The algorithmic properties of the decoupled bipartite matching heuristic is summarized as follows.

**Proposition 2** *The decoupled, bipartite matching heuristic terminates in polynomial time.*

*Proof* In the first step, the time complexity is $O((nK_{max})^3)$ as is shown in the proof of Proposition 1. The decoupled, bipartite matching heuristic terminates in at most $2d - 1$ iterations [36], in which $d = \max\{mK_{max}, n\}$. The matching process is executed at most $2d - 1$ times. The bipartite matching can be determined by the famous Hungarian algorithm, and its time complexity is $O((\max\{m, n\} \cdot K_{max})^3)$. Then the overall worst-case time complexity is $O(n^3K_{max}^3 + 2\max\{mK_{max}, n\} \cdot (\max\{m, n\} \cdot K_{max})^3)$. The proof is completed. □

## 3.4 Coupled, append heuristic

In the coupled append heuristic, new subtasks are selected, scheduled and appended to the partial task schedule iteratively. Starting from an empty schedule, subtasks with the minimal start time of all unscheduled subtasks are appended. When multiple subtasks are available, the following three priority dispatching rules can be used as tie-breakers: *STD* (shortest time difference between transport-pick robot pair), *SPT* (shortest operate/processing time), and *LPT* (longest operate/processing time). Among these rules, *SPT* and *LPT* are commonly seen in shop scheduling, and *STD* is designed specifically for the TPS problem. As the append heuristic is easy to realize, all these rules can be used simultaneously and the best schedule is chosen. For the partial schedule, the makespan objective is computed exactly by the maximum of completion times of all scheduled subtasks.

*Formal Analysis:* The following proposition summarizes the time complexity of the coupled, append heuristic.

**Proposition 3** *The coupled append heuristic terminates in polynomial time.*

*Proof* To generate a complete schedule, the append operation is run for $|\mathcal{T}| \leq m \times nK_{max}$ iterations. In the $p$-th iteration, one subtask should be chosen from $|\mathcal{T} - p|$ by a particular sorting algorithm. Generally time complexity of the sorting algorithm for $|\mathcal{T} - p|$ elements can be estimated as $O((|\mathcal{T}| - p)log(|\mathcal{T}| - p))$. To sum up, the total worst-case time complexity for the coupled, append heuristic is $O(m^2n^2K_{max}^2log(mnK_{max}))$. Then this proposition is proved. □

## 3.5 Coupled, insertion heuristic

In the coupled insertion heuristic, for the $p$-th iteration, subtasks are successively inserted into a partial sequence $\boldsymbol{S}_p$ according to a certain order, with all previous precedence relations preserved. To determine which subtask to be inserted, an insertion order can be determined by priority dispatching rules, or simply using previous heuristics. The makespan objective of a partial schedule is estimated by the maximum of the completion times of scheduled subtasks. For subtask $t_{ijk}$, let $v_i$ and $u_j$ be the numbers of scheduled subtasks of the transport robot $R_i^T$ and the pick robot $R_j^P$ respectively. To insert the subtask $t_{ijk}$ into $\boldsymbol{S}_p$, $(v_i + 1)(u_j + 1)$ possibilities exist. However, not every insertion leads to a feasible block sequence graph, because the sequence property has to be satisfied. The following three cases with inserting operations can lead to feasible solutions:

C1 Let $s_{ijk} = 1$, i.e., choose $t_{ijk}$ as the first subtask for both transport robot $R_i^T$ and pick robot $R_j^P$.

C2 Let $s_{ijk} = b + 1$, where entry $b$ occurs in the $i$-th row block or in the $j$-th column block of block rank matrix $\boldsymbol{S}_p$, i.e., subtask $t_{ijk}$ is a direct successor of one of scheduled subtasks of transport robot $R_i^T$ or pick robot $R_j^P$.

C3 If there exist a pair of subtasks $t_{ipq}$ and $t_{rjs}$ such that there does not exist any path between them in the current partial block sequence graph:

- If $r_{ipq} \leq r_{rjs}$, set $s_{ijk} = r_{rjs} + 1$ and $r_{ipq} = r_{rjs} + 2$, i.e., subtask $t_{ijk}$ is chosen as the direct successor of subtask $t_{rjs}$ and as the direct predecessor of $t_{ipq}$.
- If $r_{ipq} \geq r_{rjs}$, set $s_{ijk} = r_{ipq} + 1$ and $r_{rjs} = r_{ipq} + 2$, i.e., subtask $t_{ijk}$ is chosen as the direct successor of subtask $t_{ipq}$ and as the direct predecessor of $t_{rjs}$.
- If $r_{ipq} = r_{rjs}$, both two previous rules should be used.

In all three cases, the ranks of all successors of the inserted subtask should be updated.

*Formal Analysis:* Cases C1-C3 are the only cases that can lead to feasible sequence graph when inserting operations. The correctness and completeness of the coupled insertion heuristic are stated as the following two propositions.

**Proposition 4** *Cases C1-C3 generate all feasible sequences with the new subtask added to the current partial sequence.*

*Proof* The proof is similar to the proof of insertion operation for open shop problems in [22, 37]. Let $SR(i)$ and $SC(j)$ be the set of subtasks in row block $i$ and column block $j$ respectively, i.e., $SR(i) = \{s_{ijk}|s_{ijk} \in S\}, SC(j) = \{s_{ijk}|s_{ijk} \in S\}$. If we have a sequence on partial subtask set $S$, then we can surly construct a sequence on set $S \cup \{s_{ijk}\}$. Altogether we have $(1 + |SR(i)|)(1 + |SC(j)|)$ possibilities to include $s_{ijk}$ in the new sequence. Now we have to eliminate all cyclic cases. The number of these cases is $|SR(i)||SC(j)| - |A_1 \cup A_2|$, which means the number of new feasible sequences is:

$$\begin{aligned}&(1 + |SR(i)|)(1 + |SC(j)|) - |SR(i)||SC(j)| + |A_1 \cup A_2| \\ &\quad = 1 + |SR(i)| + |SC(j)| + |A_1| + |A_2| - |A_1 \cap A_2|. \end{aligned} \tag{4}$$

For all three cases, the number of new constructed sequences is:

$$\begin{aligned}&1 + |R \cup C| + |A_1| + |A_2| \\ &\quad = (1 + |R|)(1 + |C|) - |R \cap C| + |A_1| + |A_2| \\ &\quad = 1 + |SR(i)| + |SC(j)| + |A_1| + |A_2| - |A_1 \cap A_2|. \end{aligned} \tag{5}$$

With Eqs. (4) and (5) this proposition is proved. □

**Proposition 5** *The coupled, insertion heuristic enumerates all feasible sequences completely.*

*Proof* This proposition is obvious. From an empty task sequence, in each iteration, all feasible partial task sequences are preserved. When all subtasks are inserted, all feasible sequences are recorded. □

The following proposition shows the time complexity of the coupled, insertion heuristic.

**Proposition 6** *The coupled, insertion heuristic terminates in polynomial time.*

*Proof* The insertion operation is executed for $|\mathcal{T}| \leq mnK_{max}$ times. Now we analyze the time complexity when inserting subtask $t_{ijk}$ into a partial block rank matrix $\boldsymbol{S}_p$ in the $p$-th iteration. To update all successors of a changed subtask's rank in $\boldsymbol{S}_{p+1}$ with maximal rank $r_{p+1}^{max}$, breadth-first search should be used to analyze all its successors, and its time complexity is $O(V_p + E_p)$, in which $V_p$ and $E_p$ are the vertex set and the edge set of the partial block sequence graph $\boldsymbol{S}_p$. To obtain feasible block rank matrices with subtask $t_{ijk}$ included in the partial block rank matrix, C1 and C2 generate $(1 + v_i + u_j)$ new block rank matrices, and the time complexity is $O((1 + v_i + u_j)(V_p + E_p))$. In C3, to obtain a list of disconnected subtasks, for each of $(v_i + u_j)$ subtasks, breadth-first search should be used to analyze both its successors and predecessors, and the total time complexity is $O(2(v_i + u_j)(V_p + E_p))$. At most $(v_i + u_j)^2$ disconnected subtask pair can be obtained. for each of them, to update all successors of changed subtask's rank in $\boldsymbol{S}_{p+1}$ with maximal rank $r_{p+1}^{max}$, the time complexity is $O((v_i + u_j)^3(V_p + E_p))$. Totally for C1 to C3, the worst-case time complexity is approximated as $O((v_i + u_j)^3(V_p + E_p))$. For at most $(v_i + 1)(u_j + 1)$ new feasible sequences, to evaluate their partial makespan, the time complexity is $O((v_i+1)(u_j+1)mnK_{max})$. Finally in the $p$-th iteration, the worst-case time complexity is approximated as $O((v_i + u_j)^3(V_p + E_p) + (v_i + 1)(u_j + 1)mnK_{max})$. As the insertion operation proceeds, $v_i \to mK_{max}, u_j \to nK_{max}, V_p \to mnK_{max}, E_p \to m^2n^2K_{max}^2$. Then the previous time complexity can be rewritten as $O((m + n)^3m^2n^2K_{max}^5)$ with lower order polynomials neglected. Totally, in the worst case, the whole coupled insertion heuristic will cost $O\left((m + n)^3m^3n^3K_{max}^5\right)$ time. The proof is completed. □

The following conclusions can be drawn from previous theoretical analysis. For the coupled insertion heuristic, its computation time is highly dependent on the number of scheduled subtasks in the partial task schedule. In order to guarantee non-redundancy and feasibility of new partial sequences, complex pre-processing subroutines are required. Generation of new feasible sequences are the most time-consuming subroutine. As more and more subtasks are scheduled, the computation time for inserting new subtasks grows notably.

## 4 Performance evaluation

To evaluate the performance of these heuristics, we present computational results obtained from our MATLAB-based simulation platform. Algorithms are run on a 2.9 GHz AMD Ryzen 7-4800H PC with 16 GB RAM. The attached also video shows the simulation platform and how the heterogeneous robotic order fulfillment system works.

### 4.1 Dataset design

For TPS problems, a comprehensive, discriminative and balanced evaluation is extremely challenging: a complete enumeration of all problem constituents, including environment maps, robot kinematics, numbers of two robot types, start locations, subtask distributions, number of subtasks in one task, number of subtasks in each zone, operate times, routing-scheduling ratio, and zoning strategy is unrealistic. Pragmatically, we concentrate on the most concerned constituents, i.e., robot heterogeneity, multi-station order fulfillment tasks, and simplify other less concerned constituents. Table 2 shows instance types in our dataset. Generally, an instance can be described as $m$-$n$-$p$-$\alpha$, i.e., transport robot number, pick robot number, subtask number, and temporal-spatial ratio. For each type, we randomly generate 50 instances, and 1800 instances in total. Generation of instances are explained as follows.

#### *4.1.1 Environment*

All instances are generated on the same warehouse environment as Fig. 3 shows. In this warehouse (dimension $52\,\text{m} \times 39\,\text{m}$), 36 racks (dimension $10\,\text{m} \times 1\,\text{m}$) are arranged into a $9 \times 4$ array, thus generating $8 \times 4 = 32$ array of aisles for pick robots. These aisles can be divided equally into $n$ work zones and allocated to pick robots. The aisle width is set to 3 m, a minimal number to allow robots go through when two robots occupying opposite stations, yet not causing waste of space. The width of cross-aisles and hall area all are 2 m, a minimal number to allow robots to move in two directions. Pickup locations are adjacent to racks. Delivery locations are placed in the periphery. This map is adequate for us to generate miscellaneous instance types with different zoning strategies.

#### *4.1.2 Robots*

We assume a holonomic disc robot model with unit speed 1 m/s. As is shown in Fig. 4, to avoid collisions when one robot goes to a vertex where the other robot departs from, the diameter of the robot should be less than $\leq \frac{\sqrt{2}}{2}$ m. All robot share the same kinematics, and they are discriminated only by their task roles. In each discrete time step, a robot performs an action in the action set $\{\uparrow, \downarrow, \leftarrow, \rightarrow, \oslash\}$, i.e., move up, down, left, right, and stay still. Transport robots randomly start from delivery stations, and all pick robots start from fixed initial stations. Transport robot numbers are among 5,10,15,20,25,30. Pick robot numbers are among 8,16,32, with zoning strategies of $2\times2$ aisles,$1\times2$ aisles,$1\times1$ aisle respectively.

#### *4.1.3 Tasks*

In one task, we fix the number of subtasks as 10, a moderate task size. Comparatively in open shop scheduling problems, number of operations in each job is the machine number. However, we cannot use this rule because realistic order fulfillment tasks are surely not related to pick robots. We also notice that random numbers (e.g., uniform distribution $U(5, 15)$) are better, however, this would make it difficult to control instances, because we have allowed subtasks to be located randomly in any work zones. In other words, we stress randomness more on subtask spatial distributions rather than number of subtasks in one task. We have to make these compromises although the introduced heuristics can handle any circumstances. In one task, pickup subtasks are randomly located in pickup stations. Delivery subtasks are randomly located in delivery stations. For small ($\alpha = S$) and large ($\alpha = L$) temporal-spatial ratios, the operate time is generated subjecting to uniform distributions $U(10, 20)$ and $U(10, 50)$ respectively. Each task contains a pre-optimized subtask execution order with minimized routing components obtained by an optimal traveling salesman problem solver. Note that for instances with different pick robot numbers, we generate the same tasks, so that we can explore relations between the overall system performance and different transport-pick robot number combinations. In other words, totally 600 tasks are generated.

**Table 2** Instance Types

| | |
|---|---|
| Transport Robot Number $m$ | 5,10,15,20,15,30 |
| Pick Robot Number $n$ | $8(2\times2)$,$16(1\times2)$,$32(1\times1)$ |
| Subtask Number $p$ | 50,100,150,200,250,300 |
| Temporal-spatial Ratio $\alpha$ | S,L |

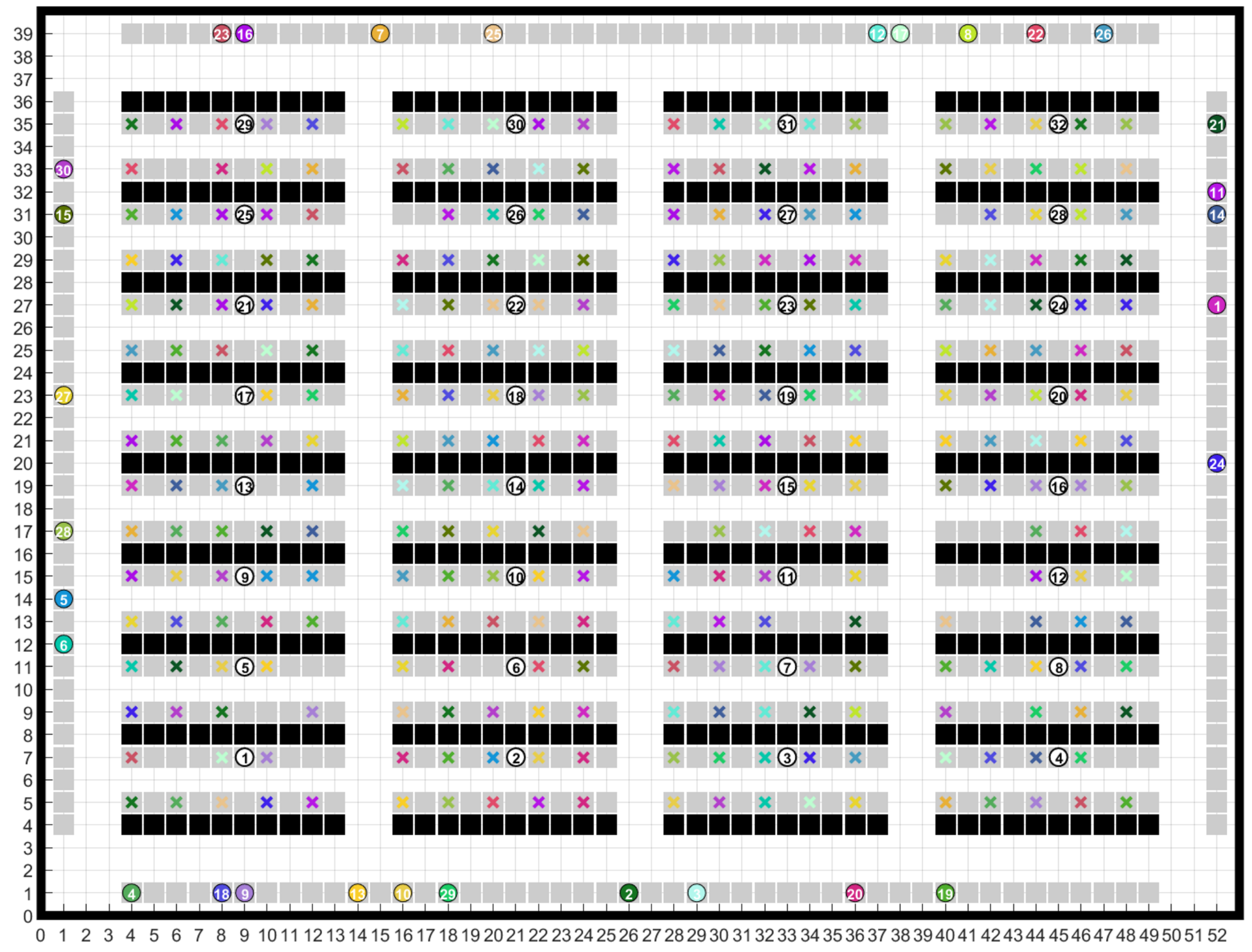

**Fig. 3** The warehouse environment in our instance design and instance 30-32-300-*S*/*L*. Black bars are racks, gray squares are pickup and delivery stations. In this instance, 30 transport robots (colored disk robots located in peripheral delivery locations) and 32 pick robots (white disk robots located in aisles) collaborate to execute 30 tasks composed of 300 subtasks (colored crosses in pickup stations)

### 4.2 Methods

The proposed constructive heuristics are abbreviated as follows:

- *DG*: Decoupled heuristic.
- *CM*: Coupled, tripartite matching heuristic.
- *DM*: Decoupled, bipartite matching heuristic.
- *CA*: Coupled, append heuristic.
- *CI*: Coupled, insertion heuristic, with insertion order generated by append heuristic.

In practice, actual travel time between two different subtask stations can be estimated by naive multiplicative robustification, i.e., multiply the ideal travel time by a factor larger than 1. Here we simply use the ideal travel time, obtained by the ideal geodesic distance between them with obstacles considered, divided by unit speed. This is feasible, because the ideal task schedule can always be scaled by the same factor and transformed to the task schedule when multiplicative robustifucation is used. For each heuristic on each instance, we record its normalized performance ratio and computation time. The normalized performance ratio obtained by a certain heuristic * is computed by $C_{max}(*)/LB$.

### 4.3 Results and analysis

First of all, for append heuristic and different priority dispatching rules, we count how many times a rule generate best value, as Table 3 shows. For instances with $n = 8$, *STD* works best; in contrast for $n = 16, 32$, *LPT* works best. This is because for smaller pick robot numbers, both pick robots' and transport robots' routing components are dominant, and *STD* rule could reduce the time difference that is required for the transport-pick robot pair rendezvous. For larger pick robot numbers, pick robots' work zones are getting smaller, and their routing component are less influential. In latter result analysis, we only

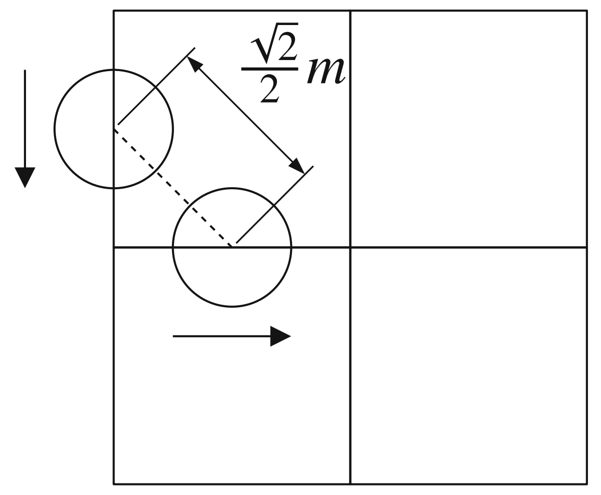

**Fig. 4** The disc robot model

choose append heuristic whose priority dispatching rule generate the best value as a representative. For $n = 8$, we choose *STD*, and for $n = 16, 32$, we choose *LPT*.

### 4.3.1 Optimality

Figure 5 shows the normalized performance ratio on different instance types. First, *DG* and *DM* generate similar results with largest performance ratio, largely because of their restrictions on problem search space. *CM* performs better than *DG* and *DM*, except for instances with $n = 32, p = 50, 100, 150$ in $m$-32-$p$-$S$ instance types and $n = 32, p = 50, 100$ in $m$-32-$p$-$L$ instance types. This is because with few subtasks, few transport robots and more pick robots, transport robots' routing components dominate the overall optimization process. In *DG* and *DM* these components are firstly minimized. For all other cases, routing and scheduling components of both robot types contribute equally to the overall optimization. The normalized performance ratio for *CA* and *CI* closely intertwine and outperform all the other heuristics regardless of problem types.

Specifically, the performance gap between coupled *CM*, *CA*, *CI* and decoupled *DG*, *DM* are more obvious on instances with relatively few pick robots ($n = 8, 16$), as pick robots' work zones are larger, and its routing subproblems are more influential. As more pick robots are used, work zones become smaller, and their routing components have a less influence on the whole problem. Extremely speaking, there is a tendency for pick robots to be stationary rather than mobile.

### 4.3.2 Computation time

We only report computation times for instances with $\alpha = S$ (results for instances with $\alpha = L$ are similar), as Table 4 shows. Computation times for *DG*, *DM*, and *CA* increase polynomially with lower-degree. For *CI*, computation times increase tremendously. It is observed that in the insertion process, as more subtasks are scheduled, more computation time is also required. For *DG*, *DM*, *CM*, and *CA*, with large pick robot numbers, more computation time is required. Oppositely for *CI*, with small pick robot numbers, more computation time is required. This is because in these instances, pick robots' work zones are larger and they have to fulfill more subtasks. The number of scheduled subtasks of pick robots grows more rapidly than the other instances. Although both append and insertion heuristics generate similar results, append is more desirable because it is simple to use and fast to compute.

### 4.3.3 Overall system performance and robot numbers

As we generate the same tasks for different pick robot numbers, we can analysis the relationship between the overall system performance and robot numbers. Take optimality results obtained by append heuristic for example. Adding more pick robots (from $n = 8$ to $n = 16$ and from $n = 16$ to $n = 32$) will definitely improve the system performance, however, the averaged improvement per pick robot will become less obvious as pick robot number increases. This is subject to the law of diminishing marginal utility [38] in economics. Practitioners can use this phenomenon to balance balance total cost

**Table 3** Best Value Frequencies for Priority Dispatching Rules

| Instance Types | *STD*[a] | *SPT*[b] | *LPT*[c] |
|---|---|---|---|
| *m*-8-*p*-*S* | **114** | 113 | 97 |
| *m*-8-*p*-*L* | **113** | 110 | 105 |
| *m*-16-*p*-*S* | 109 | 105 | **116** |
| *m*-16-*p*-*L* | 105 | 106 | **119** |
| *m*-32-*p*-*S* | 105 | 115 | **122** |
| *m*-32-*p*-*L* | 107 | 118 | **139** |

[a] *STD* shortest time difference
[b] *SPT* shortest processing time
[c] *LPT* longest processing time

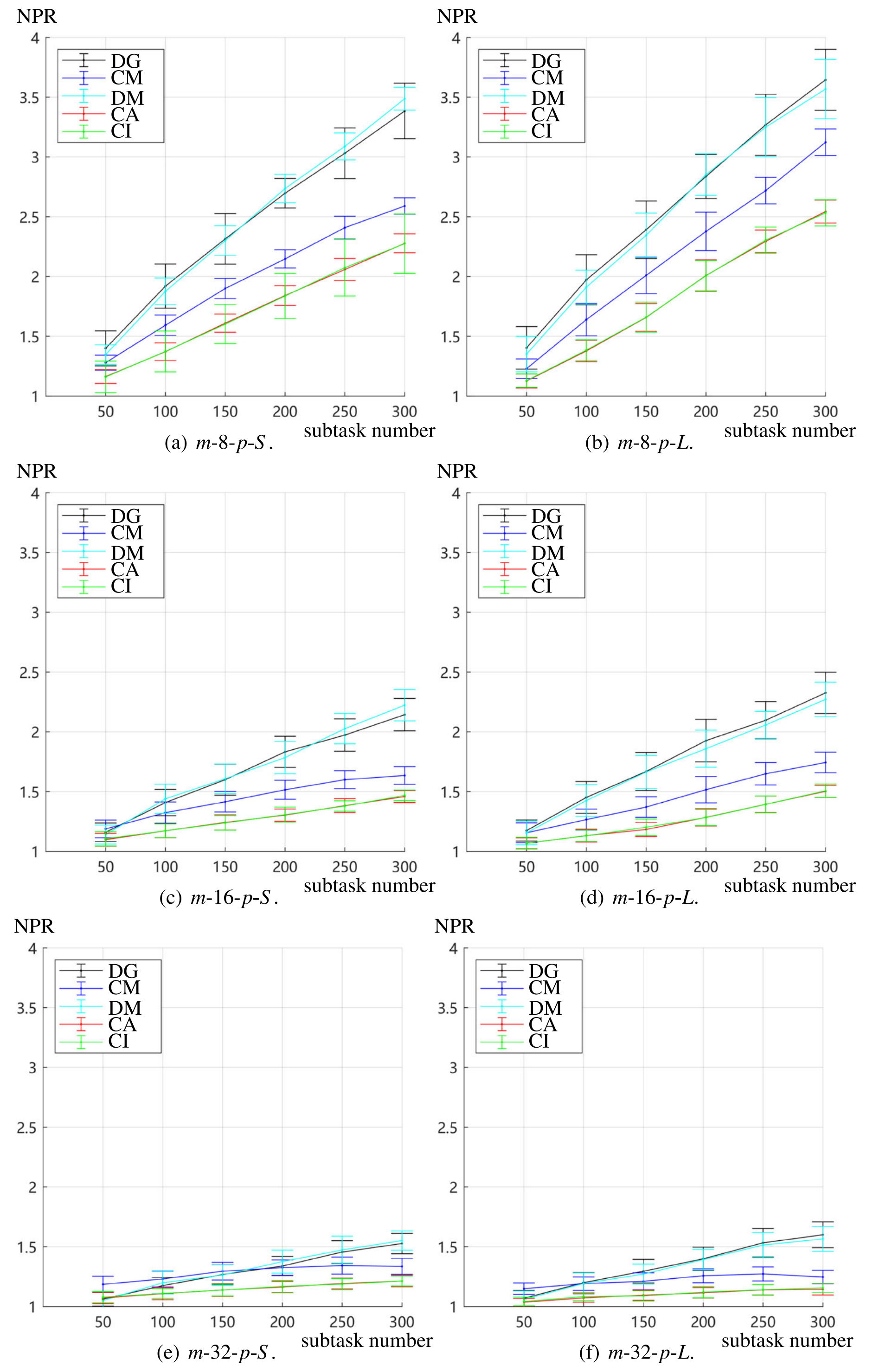

**Fig. 5** Average makespans represented by normalized performance ratio (NPR) for different instance types with different subtask numbers

(i.e., numbers of two types of robots) and throughput requirement.

*Remark* We notice that the computational results and performance analysis in this section are surely not comprehensive enough, because many other problem constituents are neglected. This is a compromise that we have to make, because it is impossible to cover all combinations. Meanwhile, currently there is also a lack of benchmarks. Nevertheless, the coupled append heuristic can

**Table 4** Average computation times for different instance types (Seconds)

| *n* | *m* | *p* | DG [a] | CM[b] | DM[c] | CA[d] | CI[e] |
|---|---|---|---|---|---|---|---|
| 32 | 5 | 50 | 0.00±0.01 | 0.38±0.05 | 0.15±0.02 | 0.92±0.11 | 14.40±1.67 |
| | 10 | 100 | 0.01±0.01 | 0.84±0.15 | 0.26±0.02 | 3.62±0.15 | 90.47±5.29 |
| | 15 | 150 | 0.03±0.01 | 1.83±0.59 | 0.34±0.03 | 7.99±0.25 | 274.90±10.65 |
| | 20 | 200 | 0.04±0.01 | 4.02±1.56 | 0.46±0.03 | 14.25±0.38 | 616.39±17.60 |
| | 25 | 250 | 0.06±0.01 | 9.20±4.14 | 0.56±0.03 | 22.45±0.61 | 1177.70±32.67 |
| | 30 | 300 | 0.09±0.01 | 30.37±12.21 | 0.70±0.06 | 32.05±1.09 | 1969.67±56.71 |
| 16 | 5 | 50 | 0.00±0.01 | 0.42±0.23 | 0.17±0.03 | 0.93±0.05 | 21.38±2.23 |
| | 10 | 100 | 0.02±0.01 | 0.86±0.10 | 0.28±0.03 | 3.47±0.10 | 140.93±8.92 |
| | 15 | 150 | 0.04±0.01 | 2.23±0.66 | 0.39±0.02 | 7.68±0.17 | 452.75±15.48 |
| | 20 | 200 | 0.07±0.01 | 6.96±2.29 | 0.55±0.03 | 14.14±0.49 | 1051.54±32.64 |
| | 25 | 250 | 0.11±0.01 | 13.12±5.04 | 0.71±0.04 | 21.83±0.62 | 1981.15±60.23 |
| | 30 | 300 | 0.17±0.03 | 18.29±4.95 | 1.02±0.13 | 33.24±2.13 | 3182.64±185.48 |
| 8 | 5 | 50 | 0.01±0.01 | 0.37±0.03 | 0.18±0.02 | 0.85±0.02 | 30.01±2.53 |
| | 10 | 100 | 0.04±0.01 | 0.99±0.10 | 0.33±0.02 | 3.41±0.07 | 230.49±8.99 |
| | 15 | 150 | 0.07±0.01 | 2.30±0.26 | 0.51±0.03 | 7.55±0.15 | 715.21±28.22 |
| | 20 | 200 | 0.12±0.01 | 4.72±0.87 | 0.76±0.03 | 13.35±0.25 | 1658.60±96.34 |
| | 25 | 250 | 0.18±0.01 | 8.24±1.67 | 0.98±0.04 | 20.87±0.41 | 3243.24±431.47 |
| | 30 | 300 | 0.25±0.01 | 13.69±3.27 | 1.29±0.05 | 29.15±0.84 | 6086.59±1854.28 |

[a] *DG* decoupled, greedy
[b] *CM* coupled, matching
[c] *DM* decoupled, matching
[d] *CA* coupled, append
[e] *CI* coupled, insertion

always be recommended for all circumstances, because of its superior optimality, and lower time complexity order. It is simple to realize, and previous partial task schedules are unchanged. □

## 5 Conclusions and future work

In this work, we conduct a comparative study on constructive heuristics using different principles for the TPS problem. Theoretically we provide formal analysis of these heuristics. Empirically, we draw the following conclusions from computational results. 1) Append heuristic is highly recommended for the TPS problem, with better performance in most cases, and within reasonable computation time. 2) Specifically, coupled heuristics work better on instances with relatively few pick robots and large work zones. 3) On the overall system performance and robot numbers, the law of diminishing marginal utility is observed.

There are many possibilities for future work. On task scheduling, we merely present several fundamental heuristics, in fact, more combined heuristics and meta-heuristics methods could also be used. Our research methodology is essentially empirical, and there is urgent requirements on developing polynomial approximate methods with bounded suboptimality. More realistic constraints could also be considered, such as task release times and due times, robots' energy constraints, robot failure. Meanwhile, online scheduling with unknown operate times and uncertain travel times is also paramount. On task execution, to enable effective and robust execution of planned task schedules, developing simultaneous task and path planning algorithms is necessary. On heterogeneous robotic order fulfillment systems, future work may include decision-making in strategic level, tactic level and operational level, including general principles on system and layout design, numbers of two types of robots, long-term system performance with time-changing workload.

**Authors' contributions**
All authors contributed to the study conception and design. Material preparation, coding, data collection and analysis were performed by Hanfu Wang. The first draft of the manuscript was written by Hanfu Wang and all authors commented on previous versions of the manuscript. Both authors read and approved the final manuscript.

**Funding**
This work is supported by the National Natural Science Foundation of China (Grant U1813206), the National Key R&D Program of China (Grant 2020YFC2007500) and the Science and Technology Commission of Shanghai Municipality (Grant 20DZ2220400).

**Availability of data and materials**
Not applicable.

**Code availability**
Not applicable.

## Declarations

**Competing interests**
The authors declared that they have no competing interests.

**Author details**
$^1$Institute of Medical Robotics, Shanghai Jiao Tong University, Shanghai 200240, China. $^2$Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China. $^3$Key Laboratory of System Control and Information Processing, Ministry of Education, Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China.

Received: 12 August 2021 Accepted: 24 November 2021
Published online: 16 December 2021

**References**

1. H. Wang, W. Chen, J. Wang, Coupled task scheduling for heterogeneous multi-robot system of two robot types performing complex-schedule order fulfillment tasks. Robot. Auton. Syst. **131**(2), 103560 (2020). https://doi.org/10.1016/j.robot.2020.103560
2. G. A. Korsah, A. Stentz, M. B. Dias, A comprehensive taxonomy for multi-robot task allocation. Int. J. Robot. Res. **32**(12), 1495–1512 (2013). https://doi.org/10.1177/0278364913496484
3. E. Nunes, M. Manner, H. Mitiche, M. Gini, A taxonomy for task allocation problems with temporal and ordering constraints. Robot. Auton. Syst. **90**, 55–70 (2017). https://doi.org/10.1016/j.robot.2016.10.008
4. B. P. Gerkey, M. J. Matarić, A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004). https://doi.org/10.1177/0278364904045564
5. Y. Zhang, L. E. Parker, in *Proceedings - IEEE International Conference on Robotics and Automation*. Multi-robot task scheduling (IEEE, 2013), pp. 2992–2998. https://doi.org/10.1109/ICRA.2013.6630992
6. M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, A. J. Kleywegt, in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Simple auctions with performance guarantees for multi-robot task allocation, vol. 1, (2004), pp. 698–705. https://doi.org/10.1109/iros.2004.1389434
7. Z. Liu, H. Wang, W. Chen, J. Yu, J. Chen, An incidental delivery based method for resolving multirobot pairwised transportation problems. IEEE Trans. Intell. Transp. Syst. **17**(7), 1852–1866 (2016). https://doi.org/10.1109/TITS.2015.2508783
8. O. Thakoor, J. Garg, R. Nagi, Multiagent UAV routing: a game theory analysis with tight price of anarchy bounds. IEEE Trans. Autom. Sci. Eng. **17**(1), 100–116 (2020). https://doi.org/10.1109/TASE.2019.2902360
9. M. Bernardine Dias, R. Zlot, N. Kalra, A. Stentz, Market-based multirobot coordination: A survey and analysis. Proc. IEEE. **94**(7), 1257–1270 (2006). https://doi.org/10.1109/JPROC.2006.876939
10. H. Wang, M. Rubenstein, Shape Formation in Homogeneous Swarms Using Local Task Swapping. IEEE Trans. Robot. (2020). https://doi.org/10.1109/TRO.2020.2967656
11. M. C. Gombolay, R. J. Wilcox, J. A. Shah, Fast scheduling of robot teams performing tasks with temporospatial constraints. IEEE Trans. Reliab. **34**(1), 220–239 (2018). https://doi.org/10.1109/TRO.2018.2795034
12. L. Garattoni, M. Birattari, Autonomous task sequencing in a robot swarm. Sci. Robot. **3**(20), 0430 (2018). https://doi.org/10.1126/scirobotics.aat0430
13. E. G. Jones, M. B. Dias, A. Stentz, Time-extended multi-robot coordination for domains with intra-path constraints. Auton. Robot. **30**(1), 41–56 (2011). https://doi.org/10.1007/s10514-010-9202-3
14. E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, A. Subramanian, New benchmark instances for the capacitated vehicle routing problem. Eur. J. Oper. Res. **257**(3), 845–858 (2017). https://doi.org/10.1016/j.ejor.2016.08.012
15. M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems, Fifth Edition*. (Springer, 2016), pp. 1–670. https://doi.org/10.1007/978-3-319-26580-3
16. I. Chernykh, A. Kononov, S. Sevastyanov, Efficient approximation algorithms for the routing open shop problem. Comput. Oper. Res. **40**(3), 841–847 (2013). https://doi.org/10.1016/j.cor.2012.01.006
17. L. R. Abreu, J. O. Cunha, B. A. Prata, J. M. Framinan, A genetic algorithm for scheduling open shops with sequence-dependent setup times. Comput. Oper. Res. **113** (2020). https://doi.org/10.1016/j.cor.2019.104793
18. G. Mejía, F. Yuraszeck, A self-tuning variable neighborhood search algorithm and an effective decoding scheme for open shop scheduling problems with travel/setup times. Eur. J. Oper. Res. **285**(2), 484–496 (2020). https://doi.org/10.1016/j.ejor.2020.02.010
19. E. Anand, R. Panneerselvam, Literature review of open shop scheduling problems. Intell. Inf. Manag. **7**(1), 33–52 (2015). https://doi.org/10.4236/iim.2015.71004
20. H. Bräsel, H. Hennes, On the open-shop problem with preemption and minimizing the average completion time. Eur. J. Oper. Res. **157**(3), 607–619 (2004). https://doi.org/10.1016/S0377-2217(03)00249-2
21. I. Averbakh, O. Berman, I. Chernykh, The routing open-shop problem on a network: Complexity and approximation. Eur. J. Oper. Res. **173**(2), 531–539 (2006). https://doi.org/10.1016/j.ejor.2005.01.034
22. H. Bräsel, A. Herms, M. Mörig, T. Tautenhahn, J. Tusch, F. Werner, Heuristic constructive algorithms for open shop scheduling to minimize mean flow time. Eur. J. Oper. Res. **189**(3), 856–870 (2008). https://doi.org/10.1016/j.ejor.2007.02.057
23. M. Andresen, H. Bräsel, M. Mörig, J. Tusch, F. Werner, P. Willenius, Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. Math. Comput. Model. **48**(7-8), 1279–1293 (2008). https://doi.org/10.1016/j.mcm.2008.01.002
24. C. Low, Y. Yeh, Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated. Robotics and Computer-Integrated Manufacturing. **25**(2), 314–322 (2009). https://doi.org/10.1016/j.rcim.2007.07.017
25. H. Wang, W. Chen, J. Wang, Heterogeneous multi-agent routing strategy for robot-and-picker-to-good order fulfillment system. Adv. Intell. Syst. Comput. **867**, 237–249 (2019). https://doi.org/10.1007/978-3-030-01370-7_19
26. A. Scholz, D. Schubert, G. Wäscher, Order picking with multiple pickers and due dates – Simultaneous solution of Order Batching, Batch Assignment and Sequencing, and Picker Routing Problems. Eur. J. Oper. Res. **263**(2), 461–478 (2017). https://doi.org/10.1016/j.ejor.2017.04.038
27. B. Naderi, S. M. T. Fatemi Ghomi, M. Aminnayeri, M. Zandieh, A contribution and new heuristics for open shop scheduling. Comput. Oper. Res. **37**(1), 213–221 (2010). https://doi.org/10.1016/j.cor.2009.04.010
28. H. Bräsel, T. Tautenhahn, F. Werner, Constructive heuristic algorithms for the open shop problem. Computing. **51**(2), 95–110 (1993). https://doi.org/10.1007/BF02243845
29. H. Bräsel, M. Harborth, T. Tautenhahn, P. Willenius, On the set of solutions of the open shop problem. Ann. Oper. Res. **92**, 241–263 (1999). https://doi.org/10.1023/A:1018938915709
30. H. Bräsel, in *Perspectives on Operations Research*. Matrices in Shop Scheduling Problems (DUV, 2007), pp. 17–41. https://doi.org/10.1007/978-3-8350-9064-4_2
31. H. Bräsel, M. Kleinau, in *System Modelling and Optimization*. On number problems for the open shop problem (Springer, 2007), pp. 145–154. https://doi.org/10.1007/bfb0113281
32. W. Yu, Z. Liu, L. Wang, T. Fan, Routing open shop and flow shop scheduling problems. Eur. J. Oper. Res. **213**(1), 24–36 (2011). https://doi.org/10.1016/j.ejor.2011.02.028
33. N. Mathew, S. L. Smith, S. L. Waslander, Planning paths for package delivery in heterogeneous multirobot teams. IEEE Trans. Autom. Sci. Eng. **12**(4), 1298–1308 (2015). https://doi.org/10.1109/TASE.2015.2461213
34. N. Kamra, T. K. S. Kumar, N. Ayanian, Combinatorial problems in multirobot battery exchange systems. IEEE Trans. Autom. Sci. Eng. **15**(2), 852–862 (2018). https://doi.org/10.1109/TASE.2017.2767379
35. M. McIntire, E. Nunes, M. Gini, in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS (AAMAS '16)*. Iterated multi-robot auctions for precedence-constrained task scheduling (International Foundation for Autonomous Agents and Multiagent Systems, Richland, 2016), pp. 1078–1086. http://dl.acm.org/citation.cfm?id=2936924.2937082
36. C. Guéret, C. Prins, Classical and new heuristics for the open-shop problem: A computational evaluation. Eur. J. Oper. Res. **107**(2), 306–314 (1998). https://doi.org/10.1016/S0377-2217(97)00332-9
37. H. Bräsel, M. Kleinau, On the number of feasible schedules of the open-shop-problem—an application of special latin rectangles. Optimization. **23**(3), 251–260 (1992). https://doi.org/10.1080/02331939208843762
38. P. Samuelson, W. Nordhaus, *Economics, McGraw-Hill Education; 19th edition (April 8, 2009)*, (2010)

## Publisher's Note