# An Adaptive Preference Joint Ant Colony System for Multi-Objective Multi-agent Pickup and Delivery

Yu-Ze Li
*College of Artificial Intelligence*
*Nankai University*
Tianjin, China

Xiao-Fang Liu
*College of Artificial Intelligence*
*Nankai University*
Tianjin, China
liuxiaofang@nankai.edu.cn

Zhi-Hui Zhan
*College of Artificial Intelligence*
*Nankai University*
Tianjin, China

Jun Zhang
*Nankai University, Tianjin, China*
*Hanyang University, ERICA, South Korea*

*Abstract*—**The multi-agent pickup and delivery (MAPD) problem aims to allocate tasks to agents and plan the path of each agent. Existing methods tend to construct solutions by selecting agents first and then tasks or opposite. They ignore the strong coupling between task allocation and path planning, meeting challenges in terms of diversity and convergence on multi-objective problems. Thus, this paper proposes an adaptive preference joint multi-objective ant colony system (APJ-MOACS) to solve bi-objective MAPD problems, i.e., minimize the total working time and balance the workload among agents. In APJ-MOACS, two pheromone matrices are built to record the historical preference between neighboring tasks in agents for two objectives. APJ-MOACS represents solutions as a sequence of agent-task pairs and constructs a joint probability distribution to achieve coordinated optimization of task selection and agent assignment. Particularly, the selection preference of an agent-task pair is determined based on four factors, i.e., pheromone, heuristic information, matching factor, and balance factor, whose weights are adaptively controlled. The matching factor describes the matching degree between agents and tasks based on load demands. The balance factor is defined based on the attribute and current state of agents to help balance workload among agents. Experimental results on 18 instances constructed based on TSPLIB demonstrate that APJ-MOACS achieves superior convergence and diversity than state-of-the-art multi-objective ant colony optimization algorithms.**

*Index Terms*—**Multi-agent pickup and delivery (MAPD), multi-objective optimization, ant colony optimization (ACO), joint probability distribution, adaptive preference.**

## I. Introduction

With the explosive growth of e-commerce and the logistics industry, the demand for goods traffic greatly increases. Multiple agents, such as unmanned aerial vehicles and autonomous vehicles, have been integrated into logistics systems to improve efficiency through collaboration. The automated scheduling of agents for task execution becomes increasingly urgent [1]. Recently, the multi-agent pickup and delivery (MAPD) problem has attracted growing attention from both academia and industry [2] [3]. MAPD problems can be seen as a variant of vehicle routing problems (VRP), which involves the task allocation and the path planning of each agent under resource constraints [4] [5]. MAPD problems are NP-Hard and pose challenges to traditional methods [6].

In the literature, multiple methods are developed to solve MAPD problems and can be broadly categorized into three classes, i.e., mathematical programming, heuristic methods, and machine learning [7]. Particularly, mathematical programming methods typically rely on mathematical modeling techniques to obtain exact or near-optimal solutions. For example, a MAPD problem is formulated as an integer linear programming or flow-optimization model to achieve global or near-optimal task allocation and path planning. MAPD problems are formulated as a minimum-cost flow model and solved by a flow-based online task assignment framework, which adopts a multi-layer flow network to represent the relationships among tasks, agents, and path feasibility for reducing the problem complexity [8]. Robotic dynamics constraints are further involved and optimized by a joint planning method, enabling precise solutions for multi-robot pickup and delivery tasks under complex physical constraints [9].

Heuristic methods employ heuristic rules or simulate intelligent behaviors to search high-quality solutions in complex search space, such as genetic algorithm, particle swarm optimization, and ant colony optimization [10]. For example, a genetic algorithm-based framework adopts a two-level chromosome encoding for task allocation and path planning to balance task coverage and total travel distance [11]. An

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62476141, in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025-00555463 & RS-2025-25456394), in part by the Tianjin Top Scientist Studio Project under Grant 24JRRCRC00030, and in part by the Tianjin Belt and Road Joint Laboratory under Grant 24PTLYHZ00250. (Corresponding authors: Xiao-Fang Liu and Jun Zhang)

adaptive large neighborhood search-based approach employs a bandit learning strategy to dynamically select neighborhood operators to improve performance in large-scale dynamic environments [12]. A collaborative parallel large neighborhood search framework is developed to enhance convergence efficiency and scalability through a multi-threaded parallel search mechanism [13]. These algorithms exhibit strong adaptability and flexibility in addressing large-scale, dynamic, and multi-objective MAPD problems.

Benefiting from the cooperative learning mechanism and pheromone-guided search strategy, ant colony system (ACS) has shown strong potential in various problems, including path planning [14] [15], task allocation [16] [17], and virtual machine placement [18]. As a representative swarm intelligence algorithm, ACS has been widely applied to MAPD and related multi-agent path planning problems [19]. However, existing methods still exhibit limitations in terms of diversity and convergence on multiobjective MAPD problems. Most methods tend to select one agent first and then one task or opposite, ignoring the coupling relationships between agents and tasks. In addition, existing methods ignore the specific attributes of heterogeneous agents, such as speed and load capacity, during task allocation and path planning, leading to imbalanced resource utilization and task allocation.

To address the above issues, this paper proposes a preference–adaptive joint multi-objective ant colony system (APJ-MOACS) to solve multi-objective MAPD problems. In APJ-MOACS, solutions are represented as a sequence of agent-task pairs and constructed by selecting agent-task pair step by step. To deal with the multi-objective issue, two pheromone matrices are built to record historical preference between tasks for the two objectives, respectively. Expect for pheromone and heuristic information, two new factors related to agent and task attributes, i.e., matching factor and balance factor, are defined to construct a joint probability distribution for agent-task pair selection. Particularly, the matching factor evaluates the matching degree of tasks to agents based on load capacity, and balance factor measures the preference of selecting an agent-task pair based on the current state of the agent and its attributes. In this way, the selection preference of agents for tasks can be adaptively adjusted to capture their interdependence. Experimental results on multiple instances with varying numbers of agents and tasks demonstrate that the proposed APJ-MOACS outperforms state-of-the-art multi-objective ACS algorithms. The main contributions of this work are as follows:

1) An adaptive preference-guided mechanism is introduced to capture the suitability of agents for tasks using a matching factor and a balance factor, which are defined based on heterogeneous agent attributes. The weights of different factors are adaptively controlled to enhance exploration in the early stages and improve exploitation in the later stages.

2) A new solution construction scheme is developed to generate solutions through selecting agent-task pairs step-by-step using a joint probability distribution. This helps to improve solution quality by utilizing the coupling relationship between agents and tasks.

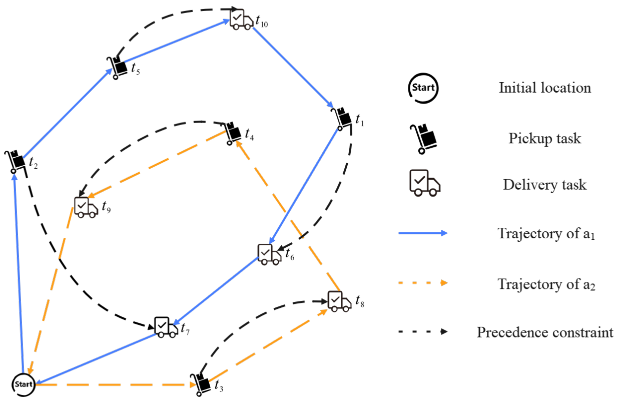


Fig. 1. An example of MAPD problem with 2 agents and 10 pickup and delivery tasks.

The paper is organized as follows. Section II formulates the bi-objective MAPD problem. Section III presents the proposed APJ-MOACS algorithm. Section IV provides experimental results. Section V concludes the paper and discusses future research directions.

## II. PROBLEM FORMULATION

Given a set of $2m$ pickup and delivery tasks $T = \{t_1, t_2, \ldots, t_{2m}\}$ and a set of $n$ heterogeneous agents $A = \{a_1, a_2, \ldots, a_n\}$, the objective is to plan a closed route for each agent, starting and ending at a common depot, to complete its assigned tasks. Fig. 1 illustrates an example of the MAPD problem. Agents $a_1$ and $a_2$ have different speeds and load capacities, and their task sequences are $S_1 = (t_2, t_5, t_{10}, t_1, t_6, t_7)$ and $S_2 = (t_3, t_8, t_4, t_9)$, respectively.

*1) Pickup and Delivery Task Definition:* Each pickup or delivery task $t_i$ is defined as a 4-tuple:

$$(ID_i, L_i, D_i, TC_i)$$

where $ID_i$ denotes the identifier of task $t_i$ $(i \in [1, 2m])$, $L_i = (x_i, y_i)$ is the coordinate of task $t_i$, $D_i$ represents the execution time of task $t_i$, and $TC_i$ is the load demand of task $t_i$. Tasks are classified into two types: pickup and delivery. For each pair of related tasks, the $k$-th pickup task $t_k$ $(k \in [1, m])$ and its corresponding delivery task $t_{k+m}$ must be executed by the same agent, and the pickup task must be completed before its corresponding delivery task.

*2) Agent Definition:* Each agent $a_j$ is represented as a 3-tuple:

$$(ID_j, V_j, AC_j)$$

where $ID_j$ is the identifier of agent $a_j$ $(j \in [1, n])$, $V_j$ denotes the velocity of $a_j$, and $AC_j$ is the maximum load capacity. Each agent can carry multiple pickups simultaneously as long as its current load does not exceed the maximum capacity. The current load $CL_j^i$ of agent $a_j$ when performing task $t_i$ is computed as:

$$CL_j^i = \begin{cases} CL_j^{-1} + TC_i, & \text{if } i \in [1, m],\ CL_j^{-1} + TC_i < AC_j \\ CL_j^{-1} - TC_i, & \text{if } i \in [m+1, 2m] \end{cases} \quad (1)$$

where $CL_j^{-1}$ represents the load of agent $a_j$ after completing the previous task.

*3) Objective Function:* The goal is to find a feasible solution that optimizes two objectives: (1) minimizing the total working time of all agents, and (2) minimizing the workload imbalance among agents.

The total working time of agent $a_i$, denoted as $T_i$, is defined as:

$$T_i = \sum_{k=1}^{|S_i|} \left( \frac{dist(L_{k-1}^i, L_k^i)}{V_i} + D_k \right) \tag{2}$$

where $|S_i|$ is the number of tasks executed by $a_i$, $L_k^i$ represents the location of the $k$-th task, $L_0^i$ is the depot location, $V_i$ is the velocity of $a_i$, and $D_k$ is the service time of task $t_k$. The Euclidean distance between two locations is defined as:

$$dist(L_i, L_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad i, j \in [1, 2m]$$

*4) Bi-objective MAPD Problem:* For a feasible solution $S$, the objective vector $F(S)$ is defined as:

$$\begin{aligned} \min F(X) &= \big(\min f_1(X) \;\; \min f_2(X)\big)^{\mathrm{T}} \\ f_1(X) &= \sum_{a_i \in A} T_i \\ f_2(X) &= \max_{a_i \in A} T_i - \min_{a_j \in A} T_j \end{aligned} \tag{3}$$

s.t.

$$\bigcup_{a_j \in A} S_j = T, \quad S_j \cap S_k = \emptyset, \quad \forall j, k \in [1, n],\ j \neq k \tag{4a}$$

$$t_k \in S_j \Leftrightarrow t_{k+m} \in S_j, \quad \forall k \in [1, m],\ j \in [1, n] \tag{4b}$$

$$\text{pos}_j(t_k) < \text{pos}_j(t_{k+m}), \quad \forall k \in [1, m],\ \text{if } t_k, t_{k+m} \in S_j \tag{4c}$$

$$CL_i^j \leq AC_j, \quad \forall i \in [1, 2m],\ j \in [1, n] \tag{4d}$$

Eq. (4a) ensures that all tasks are assigned exactly once. Eq. (4b) guarantees that paired pickup and delivery tasks are executed by the same agent. Eq. (4c) enforces the precedence constraint that each pickup must be performed before its delivery. Eq. (4d) represents the load capacity constraint of each agent.

## III. THE PROPOSED APJ-MOACS

### A. Framework of APJ-MOACS

To solve the bi-objective MAPD problem, this paper proposes a preference–adaptive joint multi-objective ant colony system (APJ-MOACS) algorithm. The proposed algorithm employs a single ant colony with two pheromone matrices to optimize two objectives simultaneously. The pheromone is deposited between tasks to record the search experience of the two objectives. At the beginning of the algorithm, an external archive is initialized as empty. In each iteration, each ant constructs a solution by selecting agent-task pair step by step until all tasks are assigned. Pheromone local update is performed to enhance solution diversity. At the end of each iteration, non-dominated solutions among all ants are collected into an external archive. The pheromone matrices are updated using high-quality solutions from the external archive.

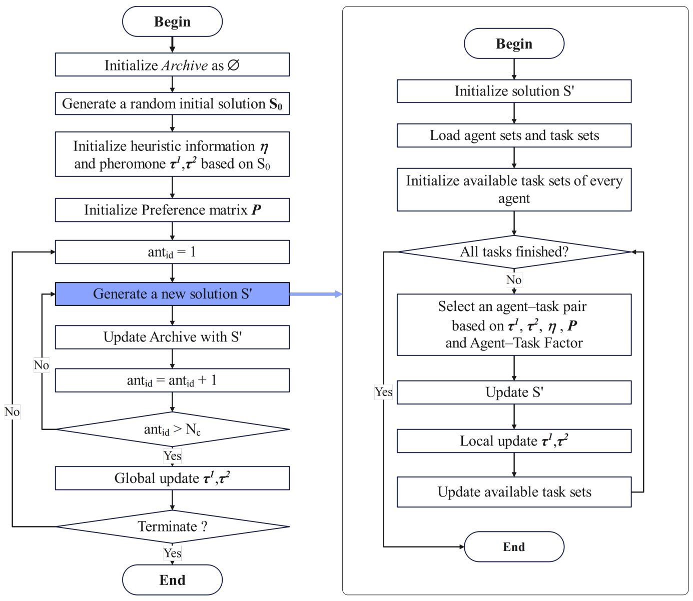

Begin
Initialize *Archive* as ∅
Generate a random initial solution $S_0$
Initialize heuristic information $\eta$ and pheromone $\tau^1$,$\tau^2$ based on $S_0$
Initialize Preference matrix $P$
$ant_{id}$ = 1
Generate a new solution S'
Update Archive with S'
$ant_{id}$ = $ant_{id}$ + 1
$ant_{id} > N_c$
No
Yes
Global update $\tau^1$,$\tau^2$
Terminate ?
No
Yes
End
Begin
Initialize solution S'
Load agent sets and task sets
Initialize available task sets of every agent
All tasks finished?
No
Select an agent–task pair based on $\tau^1$, $\tau^2$, $\eta$ , $P$ and Agent–Task Factor
Yes
Update S'
Local update $\tau^1$,$\tau^2$
Update available task sets
End

Fig. 2. Flowchart of the proposed APJ-MOACS algorithm.

The algorithm terminates until meeting termination conditions. Finally, a set of non-dominated solutions in the archive is output. The overall flowchart of the algorithm is shown in Fig. 2.

### B. Solution Representation

For the Multi-Agent Pickup and Delivery (MAPD) problem, a feasible solution $S$ can be represented as an $n$-tuple:

$$S = (S_1, S_2, \ldots, S_n)$$

where $n$ is the number of agents. Each $S_k$ denotes the task sequence assigned to agent $a_k$ $(k \in [1, n])$:

$$S_k = (t_{k_1}, t_{k_2}, \ldots, t_{k_{l_k}}), \quad t_{k_j} \in T$$

where $t_{k_j}$ is the $j$-th task performed by $a_k$, and $l_k$ is the total number of tasks assigned to $a_k$. Specifically, each task sequence $S_k$ starts and ends at the depot.

In this paper, solutions are represented as a sequence of agent-task pairs:

$$S = (u_1, u_2, ..., u_{2m})$$

where $2m$ is the number of tasks, $u_k = (a_i, t_j)$ is an agent-task pair. The solution can be decoded into multiple task sequences of agents.

### C. Solution Construction

At each iteration, each ant constructs a complete solution. Based on the solution representation, each ant iteratively expands $S$ by select one agent first and then one task for the corresponding agent until all tasks are allocated. This construction scheme narrows the search space by two steps but ignores the possibility of agent and task combinations. Thus, this paper constructs solutions by selecting feasible agent-task pairs from the remaining unassigned task set in each step. The selection of agent-task pair is based on multiple factors, including pheromone intensity, heuristic information, preference degree, and balance factors.

In $k$-th step, assume that the last task in the task sequence of each agent $a_i$ is $t_c$. For each unassigned task $t_j$, the related factors related to $a_i$ and $t_j$ are calculated to construct probability distributions for selection. Note that for agents that cannot meet the load demand of unassigned tasks, the corresponding agent-task pairs are not considered. The set of all valid agent-task pairs $(a_i, t_j)$ that satisfy capacity constraints is denoted as $\Omega$.

*1) Matching Factor:* Assigning tasks with a lower load demand to agents with smaller capacities and tasks with higher load demand to those with larger capacities helps to improve the utilization of load capacities. Since ignoring the load capacity constraint would simplify the evaluation of task-agent pair, a relaxation mechanism is introduced through a matching factor. The matching factor quantifies the degree of suitability between agents and tasks, reflecting how well a task matches an agent's capabilities.

A matching matrix $L \in \mathbb{R}^{n \times 2m}$ is defined, where each element $L_{ij}$ represents the matching degree of agent $a_i$ for task $t_j$ in term of the load capacity, which is computed as:

$$L_{ij} = \frac{1}{1 + e^{\left|\frac{TC_j}{AC_i} - \theta\right|}} \tag{5}$$

where $TC_j$ is the load requirement of the task $t_j$, $AC_i$ is the maximum load capacity of agent $a_i$, and $\theta$ is the load-matching threshold as:

$$\theta = \sqrt{\frac{TC_{\max} \cdot TC_{\min}}{AC_{\max} \cdot AC_{\min}}} \tag{6}$$

where $TC_{\max}$ and $TC_{\min}$ are the maximum and minimum load demands among all tasks, and $AC_{\max}$ and $AC_{\min}$ are the maximum and minimum load capacities among all agents, respectively.

*2) Balance Factor:* In real-world pickup-and-delivery scenarios, workload balance helps to reduce the working time and load of agents. Specially, an agent's current load state strongly influences the selection of task types, i.e., pickup or delivery. If an agent has considerable remaining capacity, pickup tasks can be assigned to improve transport efficiency; otherwise, if its current load is nearly full, delivery tasks can be assigned to help release resources.

Additionally, faster agents with higher capacity should perform more tasks to reduce the total working time. Similarly, agents with shorter working time in the current state should be prioritized to assign tasks for balancing the workloads among agents. Thus, a agent-task factor matrix $F \in \mathbb{R}^{n \times 2m}$ is defined as:

$$F_{i,j} = \begin{cases} \dfrac{V_i \times CL_i^{\text{rem}}}{T_i'}, & j \in [1, m] \\ \dfrac{V_i \times CL_i}{T_i'}, & j \in [m+1, 2m] \end{cases} \tag{7}$$

where $V_i$ denotes the speed of agent $a_i$, $CL_i$ and $CL_i^{\text{rem}}$ represent the current load and remaining load capacity of $a_i$ (with $CL_i + CL_i^{\text{rem}} = AC_i$), $AC_i$ denotes the maximum load capacity of $a_i$, and $T_i'$ represents the current total working time of $a_i$. Note that all load-related values in Eq. (7) are normalized using the maximum load capacity among all agents.

*3) Heuristic Information:* To reduce the traveling time, tasks closer to the last task for correponding agents are preferred. Thus, a heuristic information matrix $\eta \in \mathbb{R}^{(2m+1)\times(2m+1)}$ is defined between tasks, where each $\eta_{c,j}$ is calculated as:

$$\eta_{c,j} = \frac{1}{dist(L_c, L_j)} \tag{8}$$

where $dist(L_c, L_j)$ is the Euclidean distance between the locations of two tasks $t_c$ and $t_j$.

*4) Joint Probability Distribution:* Based on the pheromone, heuristic information, matching factor, and balance factor, a joint probability distribution matrix $P^{joint} \in \mathbb{R}^{n \times 2m}$ is determine as the selection probability of each agent and task pair:

$$P_{i,j}^{\text{joint}} = \begin{cases} \dfrac{(L_{i,j})^{1-\gamma}\left((\tau_{c,j}^1)^{\lambda}(\tau_{c,j}^2)^{1-\lambda}\right)^{\alpha}\eta_{c,j}^{\beta}F_{i,j}}{\sum\limits_{(c,v)\in\Omega}(L_{c,v})^{1-\gamma}\left((\tau_{c,v}^1)^{\lambda}(\tau_{c,v}^2)^{1-\lambda}\right)^{\alpha}\eta_{c,v}^{\beta}F_{c,v}}, & (a_i, t_j) \in \Omega \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where $\Omega$ is the set of all valid agent-task pairs $(a_i, t_j)$ that satisfy capacity constraints. Specifically, $L_{i,j}$ denotes the preference degree of agent $a_i$ for task $t_j$; $\tau_{c,j}^1$ and $\tau_{c,j}^2$ represent the pheromone between the current task $t_c$ being executed by agent $a_i$ and the candidate task $t_j$ under the two optimization objectives; $\eta_{c,j}$ denotes the heuristic information between tasks $t_c$ and $t_j$; and $F_{i,j}$ represents the matching degree between agent $a_i$ and task $t_j$ under the current state. Parameters $\alpha$ and $\beta$ are predefined to determine the relative importance of pheromone and heuristic information, while $\gamma$ and $\lambda$ are adaptive coefficients.

The coefficient $\gamma = I/I_{\max}$, where $I_{\max}$ is the maximum number of iterations and $1 \leq I \leq I_{\max}$ is the current iteration index. In this way, the algorithm makes use of preference information in the early stages to accelerate convergence, and gradually relies more on accumulated pheromones in later stages to avoid local optima. Similarly, $\lambda = e/(N_c + 1)$, where $N_c$ is the colony size and $1 \leq e \leq N_c$ is the current ant index, introduces behavioral differentiation among ants to enlarge the search space and improve the global exploration ability. The agent-task pair is selected according to the pseudo-random proportional rule:

$$(a_k, t_k) = \begin{cases} \arg\max_{(u,v)\in\Omega} P_{uv}^{joint}, & q \leq q_0 \\ \text{selected by } P^{joint}, & \text{otherwise} \end{cases} \tag{10}$$

where $\Omega$ is the set of avaliable agent task pairs, $q$ is a random number in a range of [0, 1], and $q_0$ is the deterministic threshold.

After the ant enters the next step to select the next agent-task pair until all tasks are assigned to agents. The solution construction procedure of an ant is presented in **Algorithm 1**.

**Algorithm 1** Solution Construction Procedure

```
Input: Set of all agents $A = \{a_1, a_2, \ldots, a_n\}$; set of all
    tasks $T = \{t_1, t_2, \ldots, t_{2m}\}$; pheromone matrices $\tau^1$,
    $\tau^2$; heuristic information matrix $\eta$; preference matrix $L$;
    feasibility factor matrix $F$
Output: A feasible solution $X_e$ for ant $e$
 1: Initialize $X_e = \emptyset$, $T' = T$
 2: while $T' \neq \emptyset$ do
 3:     Compute the joint probability $P_{ij}^{\text{joint}}$ for all feasible
        $(a_i, t_j) \in \Omega$ according to Eq. (9)
 4:     Select one feasible agent–task pair $(a_i, t_j)$ based on
        $P_{ij}^{\text{joint}}$ according to Eq. (10)
 5:     $X_{e,i} = X_{e,i} + [j]$
 6:     Update agent $a_i$'s current load $CL_i^j$ according to Eq. (1)
 7:     Local update $\tau^1$ and $\tau^2$ according to Eq. (12)
 8:     update $F$ according to Eq. (7)
 9:     $T' = T' - \{t_j\}$
10: end while
11: return The constructed feasible solution $X_e$
```

## D. Pheromone Update

*1) Pheromone Representation and Initialization:* To simultaneously optimize two objectives, two pheromone matrices $\tau^1$ and $\tau^2$ are built, each representing accumulated experience under one objective. Each $\tau^d \in \mathbb{R}^{(2m+1)\times(2m+1)}$ stores pheromone intensity $\tau_{i,j}^d$ for the transition from task $t_i$ to task $t_j$ $(d \in \{1, 2\})$. At initialization, pheromones are set using a random initial solution $X_0$:

$$\tau_0^d = \frac{1}{f_d(X_0)}, \quad d \in \{1, 2\} \tag{11}$$

and $\tau_{i,j}^d = \tau_{j,i}^d$ for all task pairs.

*2) Local Update:* After an ant select one agent-task pair, local pheromone updating is performed between the neighboring tasks $(t_i, t_j)$ in the task sequence of the corresponding agent to encourage exploration:

$$\tau_{i,j}^d = (1 - \rho_l)\tau_{i,j}^d + \rho_l \tau_0^d \tag{12}$$

where $\rho_l \in (0, 1)$ is the local evaporation coefficient.

*3) Global Update:* At the endo of each iteration, pheromone reinforcement is performed based on the non-dominated solutions stored in the external archive. For $d$-th objective, the best solution $X_b^d$ is selected from the archive, and the pheromone is updated by:

$$\tau_{i,j}^d = \begin{cases} (1 - \rho_g)\tau_{i,j}^d + \rho_g \dfrac{1}{f_d(X_b^d)}, & (t_i, t_j) \in X_b^d \\ (1 - \rho_g)\tau_{i,j}^d, & \text{otherwise} \end{cases} \tag{13}$$

where $\rho_g$ is the global evaporation rate. This strengthens paths associated with elite solutions to accelerates convergence toward optimal regions.

## IV. EXPERIMENT

### A. Experimental Setting

*1) Test Instances:* To verify the effectiveness of the proposed algorithm, multiple MAPD instances are constructed based on the classical TSP benchmark for test. Six instances are selected from the TSPLIB[1], i.e., `kroA100`, `kroA150`, `kroA200`, `kroB100`, `kroB150`, and `kroB200`. For each TSPLIB instance, three MAPD instances are generated, as summarized in Table I, in which $m$ different cities are randomly selected as the locations of pickup tasks, while another $m$ cities are taken as the locations of delivery tasks. The load demand of each task is randomly generated in a range of $[4, 10]$, and the operation time is in a range of $[10, 20]$.

In every instance, multiple agents are adopted for task execution. There are two types of agents to represent realistic operational and regulatory constraints, i.e., light-duty and heavy-duty agents. Particularly, for light-duty agents, the load capacity and velocity are randomly generated in a range of $[40, 60]$ and $[17, 20]$, respectively. For heavy-duty agents, the load capacity and velocity are randomly generated in a range of $[80, 100]$ and $[12, 15]$, respectively. All agents start at the position $(0, 0)$.

TABLE I
SETTING OF INSTANCES

| Instance | TSPLIB Instance | n | m |
|---|---|---|---|
| $I_1 - I_3$ | kroA100 | $[3, 4, 5]$ | 50 |
| $I_4 - I_6$ | kroA150 | $[3, 4, 5]$ | 75 |
| $I_7 - I_9$ | kroA200 | $[3, 4, 5]$ | 100 |
| $I_{10} - I_{12}$ | kroB100 | $[3, 4, 5]$ | 50 |
| $I_{13} - I_{15}$ | kroB150 | $[3, 4, 5]$ | 75 |
| $I_{16} - I_{18}$ | kroB200 | $[3, 4, 5]$ | 100 |

*2) Evaluation Metric:* To evaluate the performance of the algorithm, the hypervolume (MHV) is adopted, which measures the volume in the objective space dominated by the set of non-dominated solutions [20]. A larger HV value indicates a better convergence and diversity of the solution set. The objective values of all nondominated solutions are normalized using:

$$f_d'(X) = \frac{f_d(X) - f_d^{\min}}{1.1 \cdot f_d^{\max} - f_d^{\min}} \tag{14}$$

where $X$ denotes a non-dominated solution, $f_d(X)$ is the $d$-th objective value of $X$, and $f_d^{\min}$ and $f_d^{\max}$ represent the minimum and maximum values of the $d$-th objective among all algorithms. The reference point is set as $(1.0, 1.0)$.

*3) Competing Algorithms:* To demonstrate the effectiveness of the proposed algorithm on the MAPD problem, three representative multi-objective ant colony optimization algorithms are selected for comparison [21], i.e., Pareto ant colony optimization (P-ACO) [22], multiple ant colony system (MACS) [23], and multiobjective ant colony system (MOACS) [24]. Particularly, P-ACO employs two pheromone matrices and constructs solutions through stochastic weighting; MACS uses

[1]Available at: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

TABLE II
Mean HV Values of all algorithms over 20 runs

| Instance | P-ACO | MACS | MOACS | APJ-MOACS |
|---|---|---|---|---|
| $I_1$ | 0.732200±0.067667 (=) | 0.737027±0.037007 (=) | 0.719092±0.044311 (=) | 0.730001±0.040463 |
| $I_2$ | 0.514510±0.061921 (+) | 0.531722±0.064609 (+) | 0.479348±0.051608 (+) | 0.644693±0.053473 |
| $I_3$ | 0.678617±0.029102 (+) | 0.695790±0.037940 (+) | 0.716641±0.033122 (+) | 0.797889±0.031235 |
| $I_4$ | 0.675981±0.056202 (=) | 0.686186±0.044622 (=) | 0.596397±0.046353 (+) | 0.676023±0.051015 |
| $I_5$ | 0.646108±0.041288 (+) | 0.664407±0.052328 (=) | 0.686997±0.044458 (=) | 0.680271±0.030456 |
| $I_6$ | 0.501406±0.052856 (+) | 0.517276±0.061956 (+) | 0.594023±0.062515 (+) | 0.659289±0.067038 |
| $I_7$ | 0.600484±0.063175 (=) | 0.638033±0.080442 (=) | 0.555457±0.053719 (+) | 0.594980±0.062957 |
| $I_8$ | 0.711193±0.038122 (+) | 0.750592±0.036752 (=) | 0.744082±0.068406 (=) | 0.744274±0.046573 |
| $I_9$ | 0.728265±0.036991 (=) | 0.727908±0.031664 (=) | 0.706875±0.041617 (=) | 0.724869±0.041990 |
| $I_{10}$ | 0.769340±0.034257 (=) | 0.764110±0.036062 (=) | 0.755735±0.034550 (=) | 0.784705±0.056814 |
| $I_{11}$ | 0.737266±0.043707 (=) | 0.729557±0.035682 (=) | 0.698560±0.028506 (+) | 0.751410±0.042692 |
| $I_{12}$ | 0.636944±0.033408 (+) | 0.644018±0.027849 (+) | 0.678382±0.025938 (=) | 0.706985±0.047790 |
| $I_{13}$ | 0.756275±0.033366 (+) | 0.776490±0.054098 (=) | 0.773101±0.040139 (=) | 0.801500±0.043382 |
| $I_{14}$ | 0.773421±0.037805 (=) | 0.752808±0.036311 (=) | 0.646342±0.050231 (+) | 0.764318±0.051172 |
| $I_{15}$ | 0.685704±0.039176 (+) | 0.710039±0.039380 (+) | 0.667289±0.049000 (+) | 0.745762±0.037627 |
| $I_{16}$ | 0.848517±0.028304 (+) | 0.862506±0.033150 (+) | 0.858800±0.031262 (=) | 0.881899±0.033053 |
| $I_{17}$ | 0.717057±0.040983 (=) | 0.740201±0.037940 (=) | 0.726967±0.035252 (=) | 0.745846±0.034911 |
| $I_{18}$ | 0.696082±0.051144 (=) | 0.677168±0.040804 (+) | 0.622149±0.037544 (+) | 0.707340±0.039897 |
| **+ / = / -** | **9 / 9 / 0** | **7 / 11 / 0** | **9 / 9 / 0** | – |

TABLE III
Mean HV Values of APJ-MOACS and Its Variants in the Ablation Study over 20 Runs

| Instance | w/o Matching Factor | w/o Balance Factor | APJ-MOACS |
|---|---|---|---|
| $I_1$ | **0.834243**±0.046012 | 0.740327±0.036805 | 0.807629±0.033118 |
| $I_2$ | 0.618137±0.041154 | **0.633221**±0.028627 | 0.608093±0.047839 |
| $I_3$ | 0.517842±0.033032 | 0.486503±0.029839 | **0.534511**±0.023743 |
| $I_4$ | **0.776274**±0.043599 | 0.491790±0.120173 | 0.762931±0.050282 |
| $I_5$ | 0.753430±0.033458 | 0.737049±0.032706 | **0.756991**±0.032744 |
| $I_6$ | 0.692688±0.055184 | 0.481188±0.041763 | **0.707854**±0.061709 |
| $I_7$ | **0.849894**±0.048438 | 0.761539±0.047252 | 0.836061±0.045622 |
| $I_8$ | 0.668781±0.049084 | 0.635238±0.042311 | **0.671513**±0.033116 |
| $I_9$ | **0.857912**±0.022698 | 0.748348±0.037490 | 0.856387±0.031725 |
| $I_{10}$ | 0.671232±0.047045 | 0.681279±0.029245 | **0.681373**±0.052326 |
| $I_{11}$ | 0.567650±0.051610 | 0.568575±0.045401 | **0.570483**±0.048771 |
| $I_{12}$ | 0.562666±0.047525 | 0.529258±0.033280 | **0.569633**±0.045749 |
| $I_{13}$ | **0.787177**±0.041254 | 0.632032±0.075382 | 0.786765±0.044246 |
| $I_{14}$ | 0.665191±0.050137 | 0.645349±0.033229 | **0.684751**±0.043509 |
| $I_{15}$ | 0.695236±0.040918 | 0.521598±0.050474 | **0.701147**±0.033578 |
| $I_{16}$ | 0.774040±0.036670 | 0.736269±0.033893 | **0.781538**±0.039803 |
| $I_{17}$ | 0.751469±0.027805 | 0.657894±0.042453 | **0.784222**±0.035031 |
| $I_{18}$ | 0.828336±0.033154 | 0.748270±0.024445 | **0.841020**±0.037412 |
| **Best** | **5** | **1** | **12** |

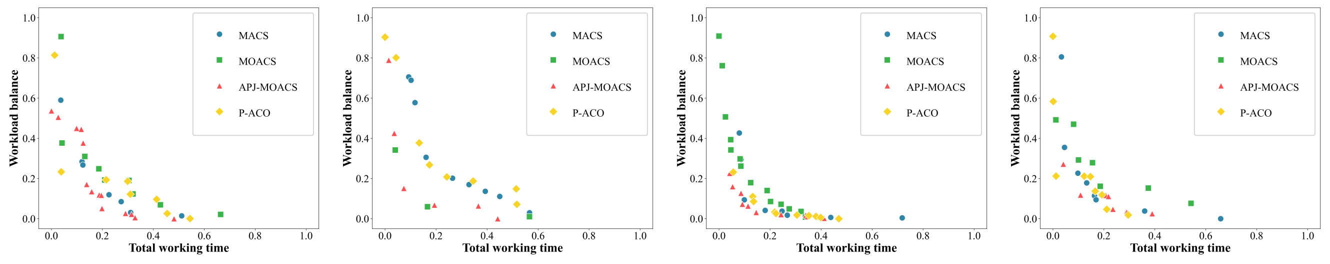
different heuristic matrices for each objective and adjusts their weights by a parameter $\gamma$; and MOACS adopts a probabilistic balancing strategy for agent selection during solution construction.

For fair comparison, all algorithms are implemented in Python with identical parameter settings, i.e., population size $= 50$, $\rho_l = \rho_g = 0.1$, $\alpha = 1$, $\beta = 2$, and $r_0 = 0.9$. The maximum function evaluation times is set as 1,000. On each instance, each algorithm independently runs 20 times and the mean HV value and standard deviation over all runs are reported. The Wilcoxon rank-sum test is employed to assess statistical differences between the proposed algorithm and each competitor at a significance level of $0.05$. The results are signed as "+" to indicate that the proposed algorithm performs significantly better, "-" to denote the significantly worse performance, and "=" to mean there is no significant difference. All experiments are conducted on a computer equipped with an Intel Core i7-13700H CPU (2.40 GHz).

## B. Experimental Results

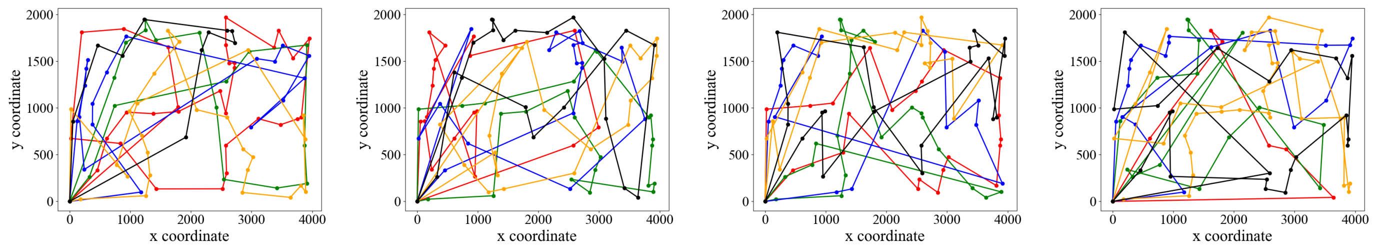
*1) Comparison with Baseline Algorithms:* The HV values of all algorithms are reported in Table II. As shown in Table II, it can be observed that the proposed APJ-MOACS algorithm achieves significantly better performance than P-ACO, MACS, and MOACS on 9 (a ratio of 50.0%), 7 (38.9%), and 9 (50.0%) out of 18 instances, respectively. Moreover, the proposed algorithm does not show any significantly worse results in any instance, indicating its stable and competitive optimization capability across different scales of MAPD instances.

The performance improvement of APJ-MOACS mainly benefits from the introduction of the adaptive preference mech-



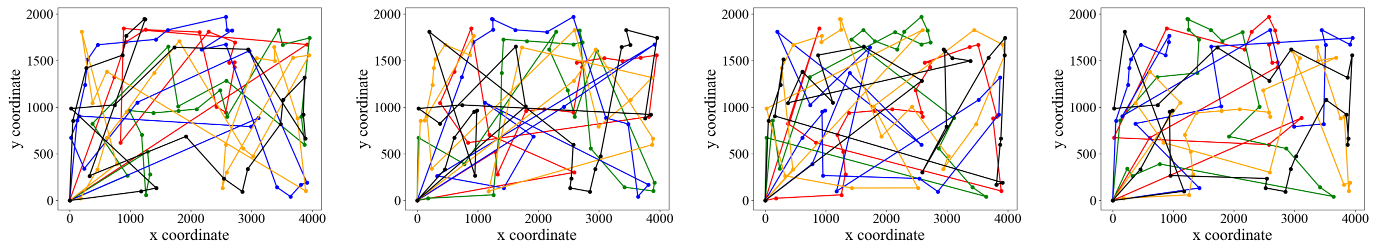
(a) (b) (c) (d)

Fig. 3. The approximated Pareto fronts and non-dominated solutions obtained by P-ACO, MACS, MOACS, and APJ-MOACS across 20 runs on (a) $I_2$, (b) $I_6$, (c) $I_{10}$, and (d) $I_{14}$.



(a) (b) (c) (d)

Fig. 4. Paths of agents with minimum total working time found by (a) P-ACO, (b) MACS, (c) MOACS, and (d) APJ-MOACS on $I_3$. Lines in different colors represent the routes of different agents.



(a) (b) (c) (d)

Fig. 5. Paths of agents with most balanced workloads found by (a) P-ACO, (b) MACS, (c) MOACS, and (d) APJ-MOACS on $I_3$. Lines in different colors represent the routes of different agents.

anism and the joint probability model. These two components dynamically balance preference information and pheromone guidance during the search process. Specially, in the early stage, higher weights are assigned to the preference based on the current state to accelerate convergence, while in the later stage, the pheromone helps to enhance global exploration. Meanwhile, the joint probability distribution directly selects agent-task pairs, helping to the coordinated and efficient task allocation during solution construction.

To further illustrate the experimental results, Fig. 3 presents the non-dominated solutions obtained by P-ACO, MACS, MOACS, and the proposed APJ-MOACS over 20 runs on instances $I_2$, $I_6$, $I_{10}$, and $I_{14}$. As shown in Fig. 3, APJ-MOACS achieves high-quality non-dominated solutions that dominate those found by other algorithms. These solutions are distributed more uniformly and closer to the approximated Pareto front, indicating the superior convergence and diversity performance of the proposed algorithm.

Taking the instance $I_3$ as an example, Fig. 4 and Fig. 5 illustrate the paths of all agents corresponding to the two minimum objective values. It can be observed that the paths generated by APJ-MOACS are overall smoother and exhibit fewer overlaps among agents compared with those of P-ACO, MACS, and MOACS. This indicates that the proposed algorithm have stronger global search and coordination capabilities.

*2) Ablation Study:* To further validate the effects of the matching factor and the balance factor in the joint probability distribution, two variants of APJ-MOACS were designed for ablation comparison, i.e., w/o Matching Factor and w/o Balance Factor. The former does not adopt the matching factor, while the latter does not adopt the balance factor. All other parameters remain consistent with the previous experiments.

As shown in Table III, the proposed APJ-MOACS algorithm achieves the highest Hypervolume (HV) values in 12 out of 18 instances (66.7%), demonstrating superior overall optimization performance. In contrast, the w/o Matching Factor and w/o Balance Factor variants achieve the best results in only 5 (27.8%) and 1 (5.6%) instances, respectively.

The w/o Matching Factor variant performs well only in a few small-scale instances with three agents, where agent–task compatibility has limited impact and pheromone guidance alone can yield satisfactory allocations. As the number of agents increases, the lack of explicit matching leads to less efficient task allocation and reduced Pareto diversity. In contrast, the w/o Balance Factor variant consistently underperforms, highlighting the critical role of the Balance Factor in maintaining solution quality across instances. Together, they form the core mechanism that enables APJ-MOACS to achieve stable and efficient multi-objective optimization performance.

## V. Conclusion

This paper proposed an adaptive preference joint multi-objective ant colony system, named APJ-MOACS, to solve bi-objective MAPD problems. Four factors, i.e., matching factor, balance factor, pheromone, and heuristic information, are adopted to build a joint probability distribution to select agent-task pairs for solution construction. Experimental results on 18 instances demonstrate that APJ-MOACS outperforms state-of-the-art multi-objective ant colony optimization algorithms. By integrating a joint probability model and adaptive parameter mechanism, the proposed algorithm achieves a dynamic balance between preference information and pheromone guidance.

In future work, we would develop multi-objective algorithms with knowledge transfer mechanisms to further improve algorithm performance [25]. In addition, reinforcement learning or neural networks can be adopted optimize pheromone trails and preference updates [26] [27].

## References

[1] X.-F. Liu, Y. Fang, Z.-H. Zhan, and J. Zhang, “Strength learning particle swarm optimization for multiobjective multirobot task scheduling,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 7, pp. 4052–4063, 2023.

[2] H. Ma, J. Li, T. Kumar, and S. Koenig, “Lifelong multi-agent path finding for online pickup and delivery tasks,” *arXiv preprint arXiv:1705.10868*, 2017.

[3] T. Bektas, “The multiple traveling salesman problem: an overview of formulations and solution procedures,” *omega*, vol. 34, no. 3, pp. 209–219, 2006.

[4] G. D. Konstantakopoulos, S. P. Gayialis, and E. P. Kechagias, “Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification,” *Operational research*, vol. 22, no. 3, pp. 2033–2062, 2022.

[5] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello, “Knapsack problems—an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems,” *Computers & Operations Research*, vol. 143, p. 105693, 2022.

[6] D. S. Hochba, “Approximation algorithms for np-hard problems,” *ACM Sigact News*, vol. 28, no. 2, pp. 40–52, 1997.

[7] J. Cai, Q. Zhu, Q. Lin, L. Ma, J. Li, and Z. Ming, “A survey of dynamic pickup and delivery problems,” *Neurocomputing*, vol. 554, p. 126631, 2023.

[8] Y. Zhang, Z. Chen, D. Harabor, P. L. Bodic, and P. J. Stuckey, “Flow-based task assignment for large-scale online multi-agent pickup and delivery,” *arXiv preprint arXiv:2508.05890*, 2025.

[9] T. Shimizu, A. Goto, K. Taneda, T. Muranaka, Y. Enoki, T. Kobayashi, T. Hattori, R. Takamido, and J. Ota, “Pickup and delivery problem solver for multiple mobile robots considering robot’s dynamics,” *ROBOMECH Journal*, vol. 12, no. 1, p. 22, 2025.

[10] V. Tomar, M. Bansal, and P. Singh, “Metaheuristic algorithms for optimization: A brief review,” *Engineering Proceedings*, vol. 59, no. 1, p. 238, 2024.

[11] A. C. L. Queiroz, H. S. Bernardino, A. B. Vieira, and H. J. Barbosa, “Solving multi-agent pickup and delivery problems using a genetic algorithm,” in *Brazilian Conference on Intelligent Systems*. Springer, 2020, pp. 140–153.

[12] T. Phan, T. Huang, B. Dilkina, and S. Koenig, “Adaptive anytime multi-agent path finding using bandit-based large neighborhood search,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 514–17 522.

[13] K. Chen, Q. Qu, F. Zhu, Z. Yi, and W. Tang, “Cplns: Cooperative parallel large neighborhood search for large-scale multi-agent path finding,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 11, pp. 2069–2086, 2024.

[14] X.-F. Liu, Y. Fang, Z.-H. Zhan, Y.-L. Jiang, and J. Zhang, “A cooperative evolutionary computation algorithm for dynamic multiobjective multi-auv path planning,” *IEEE Transactions on Industrial Informatics*, vol. 20, no. 1, pp. 669–680, 2023.

[15] K.-J. Du, J.-Y. Li, H. Wang, and J. Zhang, “A knowledge learning and random pruning-based memetic algorithm for user route planning in bike-sharing system,” *Memetic Computing*, vol. 15, no. 2, pp. 259–279, 2023.

[16] X.-F. Liu, B.-C. Lin, Z.-H. Zhan, S.-W. Jeon, and J. Zhang, “An efficient ant colony system for multi-robot task allocation with large-scale cooperative tasks and precedence constraints,” in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 1–8.

[17] T. Qian, X.-F. Liu, and Y. Fang, “A cooperative ant colony system for multiobjective multirobot task allocation with precedence constraints,” *IEEE Transactions on Evolutionary Computation*, 2024.

[18] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, “An energy efficient ant colony system for virtual machine placement in cloud computing,” *IEEE transactions on evolutionary computation*, vol. 22, no. 1, pp. 113–128, 2016.

[19] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 2002.

[20] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, “Performance assessment of multiobjective optimizers: An analysis and review,” *IEEE Transactions on evolutionary computation*, vol. 7, no. 2, pp. 117–132, 2003.

[21] R. Necula, M. Breaban, and M. Raschip, “Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems,” in *2015 IEEE 27th international conference on tools with artificial intelligence (ICTAI)*. IEEE, 2015, pp. 873–880.

[22] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer, “Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection,” *Annals of operations research*, vol. 131, no. 1, pp. 79–99, 2004.

[23] B. Barán, “A multiobjective ant colony system for vehicle routing problem with time windows,” 2003.

[24] T. Qian, X.-F. Liu, Z.-H. Zhan, J. Yong, Q. Zhang, D. Duan, C. Mao, and Y. Tang, “A multiobjective ant colony system for multi-agent pickup and delivery,” in *2024 11th International Conference on Machine Intelligence Theory and Applications (MiTA)*. IEEE, 2024, pp. 1–8.

[25] J.-Y. Li, Z.-H. Zhan, Y. Li, and J. Zhang, “Multiple tasks for multiple objectives: A new multiobjective optimization method via multitask optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 1, pp. 172–186, 2023.

[26] Z. Zong, M. Zheng, Y. Li, and D. Jin, “Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 9, 2022, pp. 9980–9988.

[27] A. ElSaid, K. Ricanek, Z. Lyu, A. Ororbia, and T. Desell, “Backpropagation-free 4d continuous ant-based neural topology search,” *Applied Soft Computing*, vol. 147, p. 110737, 2023.