IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 16, 2022, accepted March 31, 2022, date of publication April 6, 2022, date of current version June 1, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3165198

Cooperative Task Allocation for Multi-Robot Systems Based on Multi-Objective Ant Colony System

SHENGLI WANG[1], YOUJIANG LIU[2], YONGTAO QIU[2], QI ZHANG[2], FEIXIANG HUO[2], YAFAN HUANGFU[2], CHUN YANG[2], AND JIE ZHOU[2]

[1]Department of Engineering Physics, Tsinghua University, Beijing 100084, China
[2]Institute of Electronic Engineering, China Academy of Engineering Physics, Mianyang 621900, China

Corresponding author: Youjiang Liu (liuyj04@163.com)

ABSTRACT This paper proposes a novel multi-objective ant colony system (MOACS) approach to solve the cooperative task allocation problem of multi-robot systems. The task allocation problem is formulated as a multi-objective multiple traveling salesman problem (MTSP). The objectives are to minimize the total and maximum cost of the robotic vehicles so that the workload of each vehicle could be balanced. The time cost matrices of the salesmen are different and asymmetric due to the different flight speeds of vehicles and executing time of tasks. Based on the single-objective ant colony system (ACS), a novel solution construction method and a novel pheromone update rule are proposed. At each step in the solution construction phase, the ant with minimum cost has the biggest chance to add an unassigned task to balance the workload of each vehicle, while the ant with maximum cost also has a bigger chance than any other ants to add an unassigned task to find better Pareto front. The minimum value of the pheromone is limited in the pheromone update phase, which is helpful in avoiding fast convergence and local optima. Extensive simulation results suggest that the proposed MOACS has better performance and effectiveness than the existing non-dominated sorting genetic algorithm II (NSGA-II) and multi-objective particle swarm optimization (MOPSO). Hardware-in-the-loop experiments on multiple unmanned aerial vehicles (UAVs) also show that compared with NSGA-II and MOPSO, the maximum and total flight distance of the UAVs with the proposed MOACS are decreased by up to 28.46% and 26.34%, respectively, while the maximum and total time used to finish all tasks are decreased by up to 23.86% and 17.94%.

INDEX TERMS Multi-robot systems, cooperative task allocation, multi-objective optimization, multiple traveling salesmen problem, multi-objective ant colony system.

I. INTRODUCTION

Research on multi-robot systems has drawn significant interest as it is capable of performing complicated and complex tasks more effectively and efficiently, and it is more fault-tolerant compared with a single robot [1], [2]. In order to develop and deploy multi-robot systems in real-world applications, one of the most critical problems is task allocation, which aims to coordinate the robotic vehicles to perform tasks to optimize one or more objectives [3]–[5].

The existing approaches for task allocation can be categorized into decentralized and centralized ones according to the team organizational paradigm [6]. A consensus-based bundle auction (CBBA) algorithm was proposed to solve the decentralized task allocation problem [7], [8]. In CBBA, each vehicle makes its own allocation individually to maximize its local reward. Then each vehicle sends its local allocation results to the other vehicles if a communication link between them exists, as well as receiving the allocations of other vehicles and eliminating the conflicts among their allocations according to some heuristic rules. A globally consistent allocation could be achieved by repeating the above procedures. CBBA has been proven to converge within limited iterations, and 50% optimality of the final allocation can be guaranteed under diminishing marginal gain assumption. Based on the scheme of CBBA, a decentralized task allocation method named performance impact (PI) algorithm was proposed with a new concept named significance [9]. Vehicles in CBBA

The associate editor coordinating the review of this manuscript and approving it for publication was Xiwang Dong.

are selfish, while they are unselfish in PI algorithm, the objective of all vehicles is to decrease the total cost of the entire fleet as much as possible. Simulation results show that PI performs better than CBBA when applied to search and rescue scenarios with critical time constraints. Based on PI algorithm, PI soft-max algorithm was proposed in [10]. Dynamic online rescheduling was allowed in PI soft-max, and an additional soft-max action-selection procedure was introduced to increase the exploratory properties of the algorithm. Simulation results show that PI soft-max has better performance than PI algorithm.

Robustness is the main advantage of decentralized approaches. The existing vehicles can still work independently or cooperatively if some of them have failed in decentralized systems. It is also easy to add new vehicles to decentralized systems. However, it may take too much time to achieve a globally consistent allocation since too much communication is needed to eliminate the conflicts [6]. Besides, these decentralized approaches may produce highly sub-optimal solutions since locally optimal solutions may not result in globally optimal solutions, especially when the number of vehicles and tasks is large [11].

Many centralized approaches have been developed to produce solutions with better optimality. In centralized approaches, the task allocation problem is always formulated as optimization problems, which are always NP (non-deterministic polynomial)-hard. Then the formulated problems are solved with exact or heuristic methods. The results of exact methods are always the same, with the same input among different runs. The task allocation, sequencing, and scheduling problem (TASSP) was formulated as a generalization of the single traveling salesman problem (TSP), and an approximation algorithm was proposed in [12]. The multi-robot task allocation problem was formulated as an optimal assignment problem (OAP), then an exact algorithm was proposed in [13]. Simulation results show that the proposed method can find optimal assignments within reasonable time on small instances. However, it takes so much computation time to find an exact assignment of a substantial number of vehicles and tasks that these methods can hardly be applied to real-world applications.

Heuristic methods are developed to find considerable sub-optimal solutions within less computation time. The results of heuristic methods may be different even with the same input among different runs. These methods usually utilize the bionic logic or evolutionary ability to search for sub-optimal solutions [16]. These methods include genetic algorithm (GA), particle swarm optimization (PSO), wolf pack search (WPF) algorithm, and ant colony optimization (ACO) [14]–[17]. Most of these algorithms have global capability, which is helpful in dealing with NP-hard problems. However, these methods can only solve single-objective optimization problems. The objective is usually to minimize the total cost of all vehicles when applied to solve task allocation problems, the workload of each vehicle may be significantly different.

In order to balance the workload while minimizing the total cost of the vehicles, the task allocation problem has been formulated as multi-objective optimization problems (MOOPs). The task allocation problem was formulated as a multi-objective multiple traveling salesmen problem (MTSP) in [18], both the total and maximum cost of the vehicles are taken as objectives. Then a multi-objective particle swarm optimization (MOPSO) approach was proposed, its novel feature lies in a Pareto front refinement strategy and a probability-based leader selection strategy. In [19], the task allocation problem was formulated as a four-objective MOOP, and a multi-objective optimization genetic algorithm based on decision preference information (DPIMOGA) was proposed. In [20], the task allocation problem was formulated as a bi-objective MTSP, and an ant colony optimization-based metric algorithm was proposed. Computational results show that the proposed algorithm is promising and effective for the bi-objective MTSPs. However, the executing duration time of tasks is not considered in most of these works, which is necessary for real-world applications.

This paper studies the offline centralized task allocation problem of multi-robot systems, in which vehicles should balance their workload while minimizing the total costs. For online dynamic changes, the task allocation results can be modified based on the offline centralized results with the existing distributed task allocation methods, e.g., CBBA [8] and PI algorithm [9]. The task allocation problem is formulated as a multi-objective MTSP, in which the objectives are to minimize the total and maximum time cost of the vehicles. In order to find more practical solutions, another restriction is imposed on the two objectives. Then a novel multi-objective ant colony system (MOACS) approach is proposed, in which multiple pheromone and heuristic matrices are utilized to cope with multiple objectives. At each step in the solution construction phase, the ant with minimum partial cost has the biggest chance to add an unassigned task to balance the workload of each vehicle, while the ant with maximum cost also has a bigger opportunity than any other ants to add an unassigned task to obtain better Pareto front sets. Besides, a novel global pheromone update rule is proposed to bound the values of pheromone information, which is helpful in avoiding fast convergence and local optima. Extensive simulation results suggest that the solutions of the proposed MOACS can dominate those of the existing non-dominated sorting genetic algorithm II (NSGA-II) [23], [24] and MOPSO [18], and the proposed MOACS is efficient in reducing inverted generational distances (IGDs). Furthermore, experiments on multiple unmanned aerial vehicles (UAVs) were conducted on a hardware-in-the-loop platform with real onboard processing units, flight controllers, and Adhoc networks. The results show that compared with NSGA-II and MOPSO, the maximum and total flight distance of the UAVs with the proposed MOACS are decreased by up to 28.46% and 26.34%, respectively, while the maximum and total time used to finish all tasks are decreased by up to 23.86% and 17.94%.

The rest of this paper is organized as follows. In section II, the task allocation problem of multi-robot systems is described and formulated as a multi-objective MTSP, then the basic concepts of MOOPs are presented. In section III, the standard ant colony system (ACS) is introduced first, then the proposed solution construction method and pheromone update rule are presented. Simulations results are presented in section IV, which validate the proposed MOACS. Hardware-in-the-loop experimental results are presented in section V, which further show that the proposed MOACS has better performance than the existing approaches. Finally, section VI concludes this paper.

II. PROBLEM FORMULATION

A. COOPERATIVE TASK ALLOCATION PROBLEM OF MULTI-ROBOT SYSTEMS

The goal of task allocation is, given a set of N_v vehicles $\mathcal{I} = \{1, 2, \ldots, N_v\}$ and a set of N_t tasks $\mathcal{J} = \{0, 1, 2, \ldots, N_t\}$, to find a conflict-free matching of tasks to vehicles which minimizes some global costs [7]. Each task is assumed to be executed by a single vehicle, and each vehicle can perform multiple tasks in an ordered sequence. Assume that a task is assigned to more than one vehicle, then the cruising time of the vehicles that arrive at the task later than the earliest one is unnecessary, which leads to more cost. Therefore, the vehicles should perform tasks in a cooperative manner, which represents that each task needs to be assigned to at least and no more than one vehicle. Since there are N_t tasks and N_v vehicles, each task can only be performed by a single vehicle, and each vehicle can perform multiple tasks, tasks should be firstly divided into N_v subsets, each of them is performed by a vehicle. Besides, the order to perform its tasks for each vehicle should be optimized. Given the tasks being performed by a vehicle, the cost may be significantly different if the tasks are performed in different orders; thus, the order of performing tasks should be optimized to minimize the total cost.

In this paper, the local cost of each vehicle is considered as its total time used to perform tasks, which refers to the time period from when the vehicle starts to perform tasks to when it finishes its tasks and returns to base. The total cost of all vehicles is considered as the sum of local costs. Due to the different locations of vehicles and tasks, vehicles need to reach tasks before executing them, and the cruising speeds of different vehicles are assumed to be different. All vehicles start to perform tasks from the same position and return to this position after finishing all tasks. This position may be the location of a base station in real-world applications, which is named depot. In this paper, the depot is always assumed to be task $0 \in \mathcal{J}$. A duration time is lasted for a vehicle to execute a task, which refers to the time period from when the vehicle starts to execute this task to when the task is successfully executed. The duration time of a task is assumed to be different if it is executed by different vehicles due to the different capabilities of the vehicles. Therefore, the time cost

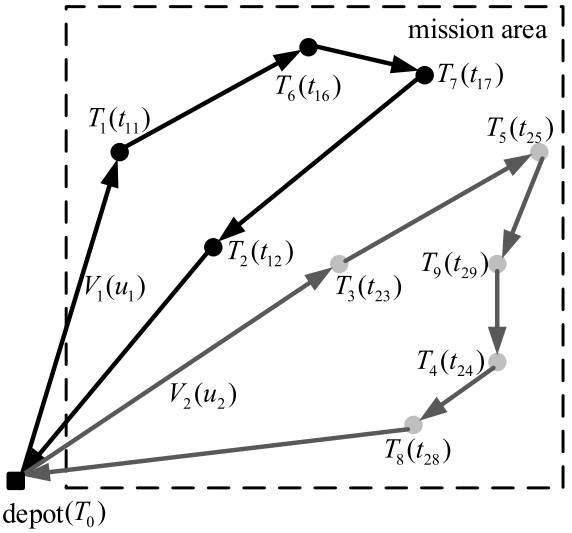


FIGURE 1. A sample task allocation problem with 2 vehicles and 9 tasks.

of a vehicle from one task to another is dependent on both the cruising time and executing duration time, and it is defined as the sum of the cruising time and the executing duration time of the former task, which is

$$c_{jk}^{(i)} = d_{jk}/u_i + t_{ij} \tag{1}$$

where $c_{jk}^{(i)}$ denotes the time cost of vehicle i from task j to k, d_{jk} is the distance from task j to k, u_i is the cruising speed of vehicle i, and t_{ij} is the duration time used by vehicle i to execute task j.

Note that the time cost defined in (1) is different for different vehicles due to the different cruising speeds of the vehicles and executing time of tasks. Besides, the time cost in (1) is usually asymmetric since $c_{jk}^{(i)} \neq c_{kj}^{(i)}$ due to the different executing duration time of task j and k. The cost is symmetric if and only if the executing time of all tasks by all vehicles is the same, i.e., $c_{jk}^{(i)} = c_{kj}^{(i)}, \forall i \in \mathcal{I}$, and $j, k \in \mathcal{J}$.

Let $\mathbf{P}_i$ be the task path of vehicle i, which is a list containing an ordered sequence of tasks in $\mathcal{J}$, its kth element $P_{ik} = j$ if vehicle i performs task j at the kth point along $\mathbf{P}_i$. Let $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{N_v}\}$ be the task paths of all vehicles. A sample task allocation problem is shown in Fig. 1, in which V_1 and V_2 represent two vehicles, while $T_1 - T_9$ are nine tasks in a mission area to be performed by the two vehicles, and the depot is outside the mission area. In this sample, the task paths of V_1 and V_2 are [0, 1, 6, 7, 2, 0] and [0, 3, 5, 9, 4, 8, 0], respectively.

Suppose that the task allocation problem is formulated as a single-objective optimization problem to minimize the total cost of all vehicles, the optimal solution is obtained when a vehicle performs all tasks if no other restrictions are imposed. If a new vehicle is used, it needs to start and end its path at the depot, thus the depot is visited many times, which leads to additional costs [25]. If only the balance of the cost of each vehicle is considered, the costs cannot be minimized, which

often leads to unnecessary costs [24]. Therefore, not only the total cost should be minimized, but also the cost of each vehicle should be balanced. To balance the workload while minimizing the total cost, besides taking the total cost as an objective, the author in [25] also took the unbalancing degree as another objective, which was measured as the difference between the maximum and minimum cost of all vehicles. However, the total cost cannot be optimized very well since it is much larger than the unbalancing degree.

In this paper, the task allocation problem is formulated as a multi-objective asymmetric MTSP, in which the objectives are to minimize the total and maximum time cost of the vehicles while the cost of each vehicle is asymmetric. The main advantage of multi-robot systems is efficiency compared with a single vehicle. Assume that the maximum time cost of the vehicles is close enough to the total time cost, then the vehicle with maximum time cost is still busy while the others have already finished their tasks, which is inefficient in reducing the time used by the vehicles to finish all tasks. Therefore, another restriction is imposed on the two objectives, which is used to measure the unbalancing degree of solutions and limit the number of solutions in the results.

The task allocation problem can be formulated as the following linear integer programming problem [24]:

$$\min \mathbf{F} = \{f_1(\mathbf{P}), f_2(\mathbf{P})\} \tag{2}$$

$$f_1(\mathbf{P}) = \sum_{i=1}^{N_v}\sum_{j=0}^{N_t}\sum_{k=0}^{N_t} x_{ijk} c_{jk}^{(i)}(\mathbf{P}_i) \tag{3}$$

$$f_2(\mathbf{P}) = \max_{1\leqslant i\leqslant N_v} \sum_{j=0}^{N_t}\sum_{k=0}^{N_t} x_{ijk} c_{jk}^{(i)}(\mathbf{P}_i) \tag{4}$$

$$s.t. \sum_{i=1}^{N_v}\sum_{j=1}^{N_t} x_{ijk} = 1, \quad \forall k \in \mathcal{J} \setminus \{0\} \tag{5}$$

$$\sum_{i=1}^{N_v}\sum_{k=1}^{N_t} x_{ijk} = 1, \quad \forall j \in \mathcal{J} \setminus \{0\} \tag{6}$$

$$\sum_{i=1}^{N_v}\sum_{j=1}^{N_t} x_{ij0} = N_v \tag{7}$$

$$\sum_{i=1}^{N_v}\sum_{k=1}^{N_t} x_{i0k} = N_v \tag{8}$$

$$\sum_{i=1}^{N_v}\sum_{j=1}^{N_t}\sum_{k=1}^{N_t} x_{ijk} = N_t \tag{9}$$

$$f_1(\mathbf{P}) \geqslant \lambda f_2(\mathbf{P}) \tag{10}$$

where x_{ijk} equals to 1 if vehicle i performs task k after finishing j and 0 otherwise, $c_{jk}^{(i)}(\mathbf{P}_i)$ is the cost of vehicle i from task j to k along task path $\mathbf{P}_i$, which is defined in (1). Equations (3) and (4) represent the total and maximum cost of the vehicles, respectively. Constraints (5) and (6) represent that each task except the depot can only be performed once, (7) and (8) represent that all vehicles start their task paths from the depot and return to the depot after finishing all tasks, while constraint (9) represents that all tasks should be performed. In (10), λ is a parameter used to limit the unbalancing degree of the task paths. The maximum value of λ is N_v when the cost of each vehicle is the same, while the minimum value is 1 when all tasks are assigned to a vehicle. In this paper, λ is set to $N_v/2$, which represents that the total cost should be bigger than $N_v/2$ times the maximum cost.

B. MULTI OBJECTIVE OPTIMIZATION PROBLEM

A MOOP with m objectives can be generally stated as:

$$\begin{aligned} &\min \mathbf{G}(\mathbf{x}) = \{g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})\} \\ &s.t.\ \mathbf{x} \in \Omega \end{aligned} \tag{11}$$

where Ω is the decision space, $\mathbf{G} : \Omega \to \mathbb{R}^m$ consists of m real-value objective functions $g_1, g_2, \ldots, g_m$, and $\mathbb{R}^m$ is called the objective space.

Let $\mathbf{u}, \mathbf{v} \in \Omega$ be two solutions to (11), $\mathbf{u}$ is said to be dominated by $\mathbf{v}$, which is denoted as $\mathbf{u} \preceq \mathbf{v}$, if $g_k(\mathbf{v}) \leqslant g_k(\mathbf{u}), \forall k = 1, 2, \ldots, m$ and $g_k(\mathbf{v}) < g_k(\mathbf{u}), \exists k = 1, 2, \ldots, m$. Similarly, $\mathbf{u} \npreceq \mathbf{v}$ denotes that $\mathbf{u}$ is not dominated by $\mathbf{v}$. A solution is said to be Pareto optimal if it is not dominated by any other solutions, while the set containing all Pareto optimal solutions is said to be Pareto front, which is denoted as $\mathcal{P}$. Hence, the goal of solving MOOPs is to find the best Pareto front sets.

III. MULTI-OBJECTIVE ANT COLONY SYSTEM

A. BASIC ANT COLONY SYSTEM

Inspired by the foraging behavior of real ants that succeed in finding the shortest paths between their nest and food sources, ACS was originally proposed by Dorigo to solve TSPs [26]. Ants lay pheromone on their paths while traveling between their nest and food sources, and are willing to select the paths with more pheromone later. Therefore, the pheromone on shorter paths accumulates faster, more and more ants will choose these shorter paths. Finally, the shortest path may be found by repeating the above procedures.

Three main phases are involved in the ACS optimization process, they are solution construction phase, local pheromone update phase, and global pheromone update phase. The artificial ants used in ACS build their solutions according to a state transition rule by iteratively adding nodes, i.e., cities in TSPs or tasks in task allocation problems, to their partially constructed solutions. In this process, two factors are taken into effect: the heuristic information about the problem being solved and the pheromone information, which is updated dynamically during the optimization process. Local pheromone update happens while ants are constructing their solutions. If an ant at a node selects another node as the next one to visit, then the pheromone on the link between the two nodes will decrease, and the link becomes less attractive to other ants. Global pheromone update happens after all ants have built their solutions. The pheromone on the links in the global best solution will increase, and these links may attract more ants in the following optimization process.

Algorithm 1 Initialization

1: $\mathcal{C} = \{1, 2, \ldots, N_t\}$ // the tasks not assigned yet
2: $\hat{P}^{(0)}_{i,1} = 0, \forall i \in \mathcal{I}$ // assign the depot to each ant
3: **while** $\mathcal{C}$ is not empty **do**
4: $\quad \hat{i} = Int(1, N_v)$ // select an ant randomly
5: $\quad r = \hat{P}^{(0)}_{\hat{i},|\hat{P}^{(0)}_{\hat{i}}|}$ // the last task of $\hat{i}$
6: $\quad s = \arg\min_{u \in \mathcal{C}} c^{(i)}_{ru}$ // find the next task
7: $\quad \hat{P}^{(0)}_{\hat{i},|\hat{P}^{(0)}_{\hat{i}}|+1} = s$ // assign s to $\hat{i}$
8: $\quad \mathcal{C} = \mathcal{C} \setminus \{s\}$ // remove s from $\mathcal{C}$
9: **end while**
10: $\hat{P}^{(0)}_{i,|\hat{\mathbf{P}}^{(0)}_i|+1} = 0, \forall i \in \mathcal{I}$ // all ants return to the depot
11: $\mathcal{P} = \{\mathbf{P}^{(0)}\}$ // $\mathcal{P}$ is initialized as $\mathbf{P}^{(0)}$
12: **for** $(r, s) \in (\mathcal{J} \cup \{0\})^2$ **do**
13: $\quad \eta^{(i)}_{rs} = 1/c^{(i)}_{rs}, \forall i \in \mathcal{I}$
14: $\quad \tau^{(1)}_{rs} = 1/f_1(\mathbf{P}^{(0)})$ // first pheromone trail
15: $\quad \tau^{(2)}_{rs} = 1/N_v/f_2(\mathbf{P}^{(0)})$ //second pheromone trail
16: **end for**

Algorithm 2 Solution Construction Method

1: **for** $j = 1$ to N_g **do**
2: $\quad \mathcal{C} = \{1, 2, \ldots, N_t\}$ // the tasks not assigned yet
3: $\quad \hat{P}_{i,1} = 0, \forall i \in \mathcal{I}$ // assign the depot to each ant
4: $\quad$ **while** $\mathcal{C}$ is not empty **do**
5: $\quad\quad$ Choose an ant $\hat{i}$ according to (12)
6: $\quad\quad r = \hat{P}_{\hat{i},|\hat{P}_{\hat{i}}|}$ // the last task of $\hat{i}$
7: $\quad\quad$ Choose the next task s according to (13)
8: $\quad\quad \hat{P}_{\hat{i},|\hat{P}_{\hat{i}}|+1} = s$ // assign s to $\hat{i}$
9: $\quad\quad$ Update pheromone on rs according to (16)
10: $\quad\quad \mathcal{C} = \mathcal{C} \setminus \{s\}$ // remove s from $\mathcal{C}$
11: $\quad$ **end while**
12: $\quad \hat{P}_{i,|\hat{P}_i|+1} = 0, \forall i \in \mathcal{I}$ // all ants return to depot
13: $\quad \mathcal{P} = \mathcal{P} \cup \{\mathbf{P}\}$, if $\mathbf{P} \npreceq \mathbf{P}^{(l)}, \forall \mathbf{P}^{(l)} \in \mathcal{P}$
14: $\quad$ **for** $l = 1$ to $|\mathcal{P}|$ **do**
15: $\quad\quad \mathcal{P} = \mathcal{P} \setminus \{\mathbf{P}^{(l)}\}$, if $\mathbf{P}^{(l)} \preceq \mathbf{P}$
16: $\quad$ **end for**
17: **end for**

B. MULTI-OBJECTIVE ANT COLONY SYSTEM

1) BASIC SETTINGS AND INITIALIZATION

In order to cope with multiple salesmen and objectives, the data structures of solutions, ant groups, pheromone and heuristic information matrices, and Pareto front in the proposed MOACS are first stated as follows. Each solution $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{N_v}\}$ of the multi-objective MTSP contains N_v ordered sequences of tasks, the ith sequence $\mathbf{P}_i$ is the task path of vehicle i. Each ant group constructs a solution at each iteration, and N_v ants are employed in each group. For a link between task r and s, there are two kinds of pheromone $\tau^{(1)}_{rs}$ and $\tau^{(2)}_{rs}$. Since the time cost of each vehicle from one task to another is different, the heuristic information of each ant is different. N_v heuristic matrices are employed, and the ith heuristic information between task r and s for the ith ant in a group is denoted as $\eta^{(i)}_{rs}$. The Pareto front $\mathcal{P}$ contains all non-dominated solutions found so far, and its lth element $\mathbf{P}^{(l)}$ is a non-dominated solution to the multi-objective MTSP.

The procedures of initialization are summarized as Algorithm 1. In order to obtain the initial value of the pheromone matrices, a feasible solution $\mathbf{P}^{(0)}$ is generated as follows. An ant in a group is randomly selected firstly, and the next tasks being assigned to this ant is chosen as the closest unassigned one relative to the last task of this ant according to the time cost (Line 1-10 in Algorithm 1, $\hat{P}^{(0)}_i$ is the ith partially constructed task path of $P^{(0)}$). The above procedures repeat until all tasks have been assigned to the ants in a group.

The Pareto front set is initialized as $\{\mathbf{P}^{(0)}\}$, and the heuristic information on a link between task r and s for the ith ant in a group is initialized as the inverse of the time cost, i.e. $\eta^{(i)}_{rs} = 1/c^{(i)}_{rs}$. Then the initial value of the pheromone matrices for the first objective is set to $\tau^{(1)}_0 = 1/f(\mathbf{P}^{(0)})$. Since f_1 is approximately N_v times of f_2 in the most ideal case and the two pheromone matrices should be balanced, for the second objective, the initial value of the pheromone is set to $\tau^{(2)}_0 = 1/N_v/f_2(\mathbf{P}^{(0)})$.

2) SOLUTION CONSTRUCTION

The procedures of the solution construction phase are summarized as Algorithm 2. At the beginning of each iteration, N_g ant groups are deployed at the depot, and each ant group builds a solution during each iteration. Since there are N_v ants in each group and tasks are iteratively added to the ants one by one, an ant needs to be first selected to add an unassigned task. The partial cost of each ant in an ant group could be obtained at each step of the solution construction phase because tasks are assigned one by one. On the one hand, in order to minimize the maximum cost of all ants in a group, the ant with the minimum partial cost is selected to add an unassigned task. In this way, only the minimum partial cost increases, then the maximum cost could be minimized, and the cost of each ant in a group could also be balanced. On the other hand, if only the ant with the minimum partial cost is selected, the algorithm will converge within very few iterations and be trapped into local optimal solutions. Therefore, the ant with maximum partial cost also has a bigger chance to be selected to avoid local optima. Since the optimal solution is obtained once an ant in a group performs all tasks without additional restrictions, more non-dominated solutions may be found by selecting the ant with maximum partial cost. Besides, the ants except those with minimum and maximum cost should also have chances to be selected to add a new task. Therefore, the ant $\hat{i}$ adding the next unassigned task is selected as follows.

$$\hat{i} = \begin{cases} \arg\min_{1 \leqslant i \leqslant N_v} C(\hat{\mathbf{P}}_i), & \text{if } q < q_0 \\ \arg\max_{1 \leqslant i \leqslant N_v} C(\hat{\mathbf{P}}_i), & \text{if } q > 1 - q_1 \\ Int(1, N_v), & \text{otherwise} \end{cases} \tag{12}$$

where $\hat{\mathbf{P}}_i$ is partially constructed solution of ith ant in a group, q is a random number uniformly distributed in [0, 1]. q_0 is the

probability that the ant with minimum partial cost is selected, while q_1 is the probability that the ant with maximum partial cost is selected. $Int(1, N_v)$ is a random integer between 1 and N_v, and $C(\hat{\mathbf{P}}_i)$ is the partial cost of ith ant, which is defined as

$$C(\hat{\mathbf{P}}_i) = \sum_{j=0}^{N_t} \sum_{k=0}^{N_t} x_{ijk} c_{jk}^{(i)}(\hat{\mathbf{P}}_i) \tag{13}$$

Once the ant $\hat{i}$ in a group is selected, let r be the last task in $\hat{\mathbf{P}}_{\hat{i}}$, and task s being assigned to $\hat{i}$ is selected according to the following state transition rule:

$$s = \begin{cases} \arg\max_{u \in \mathcal{C}} [\prod_{k=1}^{2} [\tau_{ru}^{(k)}]^{\alpha_k}] \cdot [\eta_{ru}^{(\hat{i})}]^{\beta}, & \text{if } p < p_0 \\ S, & \text{otherwise} \end{cases} \tag{14}$$

where $\mathcal{C}$ is the set that contains the tasks that have not been assigned yet, α_1, α_2 and β are three parameters, which are used to determine the relative importance of pheromone versus heuristic information. p is a random number uniformly distributed in [0, 1], and $0 \leqslant p_0 \leqslant 1$ is a parameter to determine which way is selected to obtain the next task to be assigned. Here, $[\prod_{k=1}^{2} [\tau_{ru}^{(k)}]^{\alpha_k}] \cdot [\eta_{ru}^{(\hat{i})}]^{\beta}$ is said to be the decision information of task u, which is used to decide the next task being assigned. If $p < p_0$, the task with maximum decision information is assigned. S is a random variable selected according to the following probability distribution.

$$p_{rs} = \begin{cases} \dfrac{[\prod_{k=1}^{2} [\tau_{rs}^{(k)}]^{\alpha_k}] \cdot [\eta_{rs}^{(\hat{i})}]^{\beta}}{\sum_{u \in \mathcal{C}} [\prod_{k=1}^{2} [\tau_{ru}^{(k)}]^{\alpha_k}] \cdot [\eta_{ru}^{(\hat{i})}]^{\beta}}, & \text{if } s \in \mathcal{C} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Equation (15) represents that if $p \geqslant p_0$, the next task being assigned is selected randomly from the unassigned ones according to their decision information. The task with bigger decision information has a bigger chance to be selected. Once a task s is selected to be assigned to $\hat{i}$, it will be removed from the unassigned set $\mathcal{C}$.

The above solution construction procedures repeat until all tasks have been assigned. Then the depot, i.e., task $0 \in \mathcal{J}$, is assigned to each ant in the group. If the newly constructed solution is not dominated by any solutions in the Pareto front set, i.e., $\mathbf{P} \not\preceq \mathbf{P}^{(l)}, \forall \mathbf{P}^{(l)} \in \mathcal{P}$, it will be added to the Pareto set (Line 13 in Algorithm 2). The solutions in the Pareto set may be dominated by this newly constructed solution, and the dominated ones are removed from the Pareto set(Line 14-16 in Algorithm 2).

3) PHEROMONE UPDATE

Each time an ant at task r in a group selects s as the next one being assigned, the local pheromone update rule is applied on the link between task r and s. The pheromone level is updated as

$$\tau_{rs}^{(k)} \leftarrow (1 - \rho) \cdot \tau_{rs}^{(k)} + \rho \cdot \tau_0^{(k)}, \quad k = 1, 2 \tag{16}$$

where $0 \leqslant \rho \leqslant 1$ is a parameter used to balance the historical and newly added pheromone.

Algorithm 3 Global Pheromone Update Rule

```
for (r, s) ∈ (J ∪ {0})^2 do
    Δτ_rs^(k) = τ_0^(k) // initialize Δτ_rs^(k)
end for
for l = 1 to |P| do
    for i = 1 to N_v do
        for j = 1 to |P_i^(l)| − 1 do
            r = P_{i,j}^(l), s = P_{i,j+1}^(l)
            Update Δτ_rs^(k) according to (18)
        end for
    end for
end for
for (r, s) ∈ (J ∪ {0})^2 do
    Update pheromone on rs according to (17) and (18)
end for
```

The global pheromone update phase, which is summarized as Algorithm 3, happens after all ant groups have built their solutions during each iteration. The pheromone level on the link between task r and s is updated as

$$\tau_{rs}^{(k)} \leftarrow (1 - \rho) \cdot \tau_{rs}^{(k)} + \rho \cdot \Delta\tau_{rs}^{(k)}, \quad k = 1, 2 \tag{17}$$

where $\Delta\tau_{rs}^{(k)}$ is the newly added pheromone on the link between r and s for the kth objective, which is defined as

$$\Delta\tau_{rs}^{(k)} = \begin{cases} \tau_0^{(k)} + \sum_{l=1}^{|\mathcal{P}|} \dfrac{1}{n_k \cdot f_k(\mathbf{P}^{(l)})}, & \text{if } rs \in \mathbf{P}^{(l)}, \mathbf{P}^{(l)} \in \mathcal{P} \\ \tau_0^{(k)}, & \text{otherwise} \end{cases} \tag{18}$$

where $\mathbf{P}^{(l)}$ denotes the lth non-dominated solution in the Pareto front set, $|\mathcal{P}|$ is the number of solutions in $\mathcal{P}$, n_k is a parameter used to balance the two pheromone matrices, $n_k = 1$ if $k = 1$ and $n_k = N_v$ if $k = 2$.

All the solutions in $\mathcal{P}$ are not dominated by $\mathbf{P}^{(0)}$, then for any $k = 1, 2$, there exists $\mathbf{P}^{(l)} \in \mathcal{P}$, which satisfies $f_k(\mathbf{P}^{(l)}) \leqslant f_k(\mathbf{P}^{(0)})$, thus $\Delta\tau_{rs}^{(k)} \geqslant \tau_0^{(k)}$ always holds. $\Delta\tau_{rs}^{(k)} = \tau_0^{(k)}$ if and only if the solutions which is not dominated by $\mathbf{P}^{(0)}$ have not been found yet, and only $\mathbf{P}^{(0)}$ is in $\mathcal{P}$. Therefore, the pheromone level on the links in the solutions in $\mathcal{P}$ increases by applying the global pheromone update rule, and more ants are attracted to select these links to find better solutions.

For a link between r and s, if it has never been on the solutions in $\mathcal{P}$, the pheromone level for the kth objective is always $\tau_0^{(k)}$. When the local pheromone update rule is applied, the pheromone level is still $\tau_0^{(k)}$. If the link between task r and s has ever been on the solutions in the global Pareto front set, the pheromone level will be higher than $\tau_0^{(k)}$ if $\rho \neq 0$. When the local pheromone update rule is applied, the pheromone level will decrease, but $\tau_{rs}^{(k)} \geqslant \tau_{rs}^{(0)}$ always holds. Therefore, the minimum pheromone level is limited by $\tau_0^{(0)}$ with the proposed global pheromone update rule, which is useful in avoiding fast convergence and local optima. The goal of local pheromone update is to decrease the pheromone on the links that are just selected by an ant, and the attractiveness of this

link to other ants decreases, so that the ants in different groups may not select the same path repeatedly.

C. COMPUTATIONAL COMPLEXITY

In the initialization phase, the computational complexity of constructing a feasible solution $\mathbf{P}^{(0)}$ (line 1-10 in Algorithm 1) is $O(N_t^2)$ in the worst case, while the complexity of the initialization (line 11-16 in Algorithm 1) is $O((N_t+1)^2)$. Therefore, the total complexity of the initialization phase is summed to be $O(N_t^2)$.

In the solution construction phase, the computational complexity of constructing a new solution (line 2-12 in Algorithm 2) is $O(N_v N_t^2)$, while the complexity of removing dominated solutions in $\mathcal{P}$ (line 13-16 in Algorithm 2) is $O(|\mathcal{P}|)$. Since there are N_g ant groups, the solution construction phase repeats N_g times. Therefore, the total complexity of solution construction phase is summed to be $O(N_g(N_v N_t^2+|\mathcal{P}|))$.

In the global pheromone update phase, the computational complexity of calculating new added pheromone $\Delta\tau_{rs}^{(k)}$ (line 1-11 in Algorithm 3) is $O(N_v N_t|\mathcal{P}|)$ in the worst case, while the complexity of updating pheromone (line 12-14 in Algorithm 3) is $O((N_t+1)^2)$. Therefore, the total complexity of global pheromone update is summed to be $O(N_t^2+N_v N_t|\mathcal{P}|)$.

Assume that the maximum number of iterations is set to N, then the solution construction phase and pheromone update phase repeat N times, and the complexity of initialization is much less than that of the solution construction phase. Therefore, the total complexity of the proposed MOACS is bounded by $O(N(N_g N_v N_t^2+N_v N_t|\mathcal{P}|+N_g|\mathcal{P}|))$.

IV. SIMULATIONS AND RESULTS

A. PARAMETER CONFIGURATIONS

The proposed approach is simulated on several different scenarios in which the tasks to be performed are assumed to be known. Since there are currently no benchmark examples for multi-objective MTSPs to be used to evaluate the performance of various approaches, most existing work is done on the TSP benchmark [19]. In this paper, four instances in TSPLIB[1] named kroA100, kroA150, kroA200, and kroB150 were tested. The number at the last of the instance name represents the number of positions listed in the instance, e.g., there are 100 and 150 positions in kroA100 and kroA150, respectively. All vehicles were assumed to be at the first position in the instances before performing tasks, while the tasks to be performed were assumed to be at the other positions. The units of the positions given in the TSP instances were assumed to be meters. The cruising speeds of the vehicles were uniformly set between 20 m/s and 30 m/s, while the executing duration time of a task performed by a vehicle was uniformly set between 50 s to 100 s.

The proposed MOACS was compared with the existing well-known NSGA-II and MOPSO. NSGA-II extends GA to cope with multiple objectives. At each iteration, a new temporary population is generated by crossover and mutation operators, the operators in [24] were adopted in this paper. Then the individuals in the union of this temporary population and the original population are sorted by their dominated level and crowding distances [23]. The individuals with lower dominated levels and bigger crowded distances are accepted to the next generation, while the others are rejected. Therefore, the optimality of solutions increases along with iterations. PSO was proposed in [27] to train the artificial neural network weights in 1995 and has been successfully applied in many optimization problems. At each iteration, the velocity of each particle is updated with its current velocity, position, local and global best positions. Then the position of each particle is updated with the velocity that is just updated. The local and global best positions are updated if the new position is better than them. The author in [18] extends the standard single-objective PSO to solve the multi-objective MTSP by proposing a Pareto front refinement strategy and a probability-based leader selection strategy. Besides, a Hamiltonian tour improvement algorithm is utilized to improve the task path of each vehicle to decrease the costs further.

TABLE 1. Detailed values of the parameters used in the proposed MOACS.

Notation	Description	Value
N_g	Number of ant groups	24
q_0	Probability of selecting minimum-partial-cost ant	0.90
q_1	Probability of selecting maximum-partial-cost ant	0.05
α_1	Exponent of the first pheromone matrix	1
α_2	Exponent of the second pheromone matrix	1
β	Exponent of the heuristic information	2
p_0	Probability of selecting the task with maximum decision information	0.9
ρ	Relative importance of the newly added and historical pheromone information	0.5

The parameters used in the proposed MOACS are shown in Table 1. In order to make a fair comparison, the number of populations in NSGA-II and the number of particles in MOPSO were also set to 24. The other parameters used in NSGA-II and MOPSO were the same as those in [24] and [18], respectively. For all three algorithms, the number of iterations was set to 100. To evaluate the obtained Pareto fronts, 20 independent runs for each investigated algorithm and TSP instance were performed. From these solutions obtained among 20 runs, the non-dominated ones were extracted to obtain a single approximate Pareto front set for each algorithm and each MTSP instance.

B. SIMULATION RESULTS

Fig. 2 depicts the approximate Pareto front sets obtained by the three investigated algorithms on the instance named kroB150 with a different number of vehicles. The horizontal and vertical axis represents the total and maximum time of the vehicles used to perform all the tasks. The time used by a vehicle to perform its tasks refers to the time period from when it starts to perform tasks to when it finishes

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

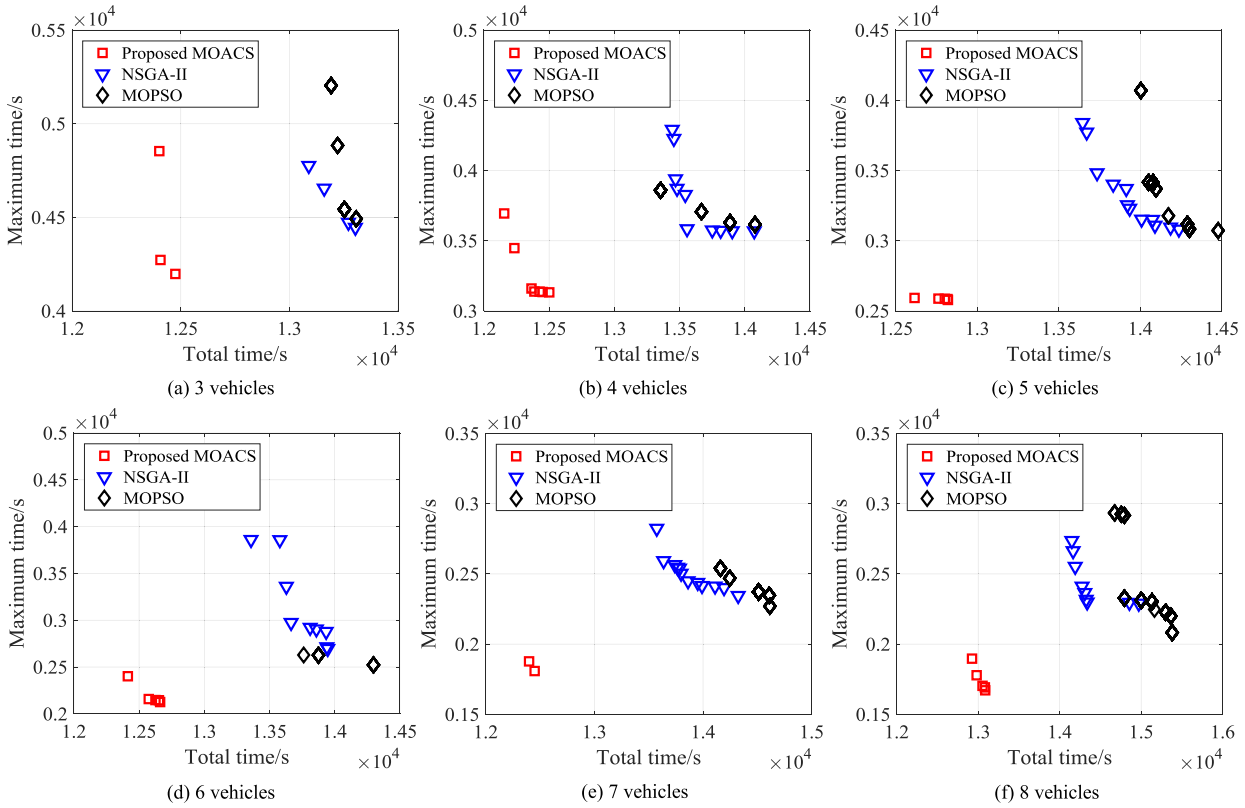


FIGURE 2. Non-dominated fronts obtained by the proposed MOACS, NSGA-II and MOPSO on kroB150 with a different number of vehicles.

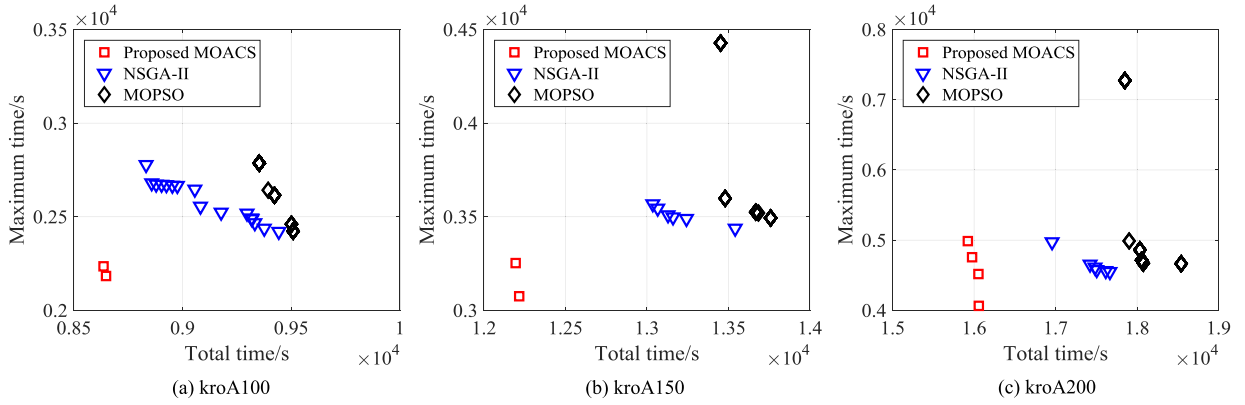


FIGURE 3. Non-dominated fronts obtained by the proposed MOACS, NSGA-II and MOPSO on kroA100, kroA150 and kroA200 with 4 vehicles.

all its tasks and returns to the depot. As shown in the figure, the solutions of the proposed MOACS always dominate those of NSGA-II and MOPSO despite the number of vehicles. Furthermore, the three algorithms were evaluated with different TSP instances named kroA100, kroA150, and kroA200, and the number of vehicles was set to 4. The approximate Pareto front sets obtained by each algorithm are shown in Fig. 3. The solutions of the proposed MOACS for all three instances also dominate those of NSGA-II and MOPSO.

Note that the Pareto sets of the proposed MOACS are more sparse than NSGA-II and MOPSO. One reason is that in order to balance the workload of each vehicle, some solutions are excluded by applying the constraint (10). Another reason is that the two objectives are not negatively correlated. One objective will decrease as the other one increase. On the contrary, the objectives are positively correlated to some degree. The decrease in the maximum time may also result in a decrease in the total time. An ideal case is that the costs of all vehicles are the same and minimized, then only one

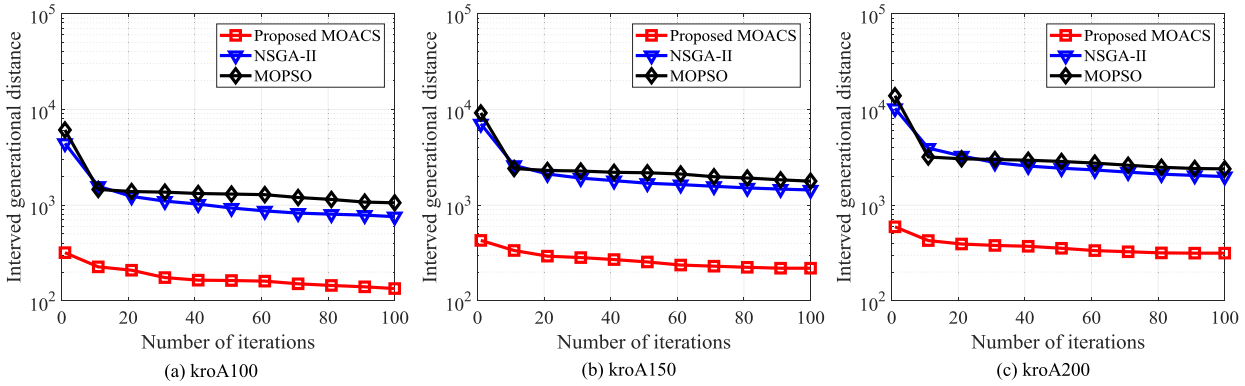


FIGURE 4. Evolution of average IGD values along with the number of iterations on instances named as kroA100, kroA150 and kroA200.

TABLE 2. Average CPU time used by the proposed MOACS, NSGA-II and MOPSO after 100 iterations (unit: seconds).

Instance		MOACS		NSGA-II		MOPSO	
Name	N_v	mean	std. dev.	mean	std. dev.	mean	std. dev.
kroB150	3	5.33	0.13	14.44	0.32	50.38	23.49
kroB150	4	5.32	0.06	16.67	0.50	56.94	41.08
kroB150	5	5.26	0.07	18.66	0.59	51.15	29.02
kroB150	6	5.36	0.08	21.41	0.72	65.83	56.52
kroB150	7	5.27	0.08	23.17	0.96	76.40	65.82
kroB150	8	5.34	0.16	24.77	0.89	72.11	37.74
kroA100	4	3.47	0.06	15.52	0.88	46.28	33.76
kroA150	4	5.58	0.10	21.06	0.60	53.79	27.95
kroB200	4	7.29	0.14	17.86	0.49	54.49	27.50

solution is in the Pareto front set. Even though the Pareto front set of the proposed MOACS is more sparse, the solutions of the proposed MOACS can still dominate those of NSGA-II and MOPSO. The proposed MOACS is more practical since only one solution is needed in most real-world applications.

The algorithms were implemented with MATLAB 2017b on a computer with a CPU Intel Core i5-7500 of 3.4 GHz and 8 GB of RAM. Table 2 shows the average and standard deviation of CPU time used by each algorithm. Among the three algorithms, the proposed MOACS takes the least CPU time, while MOPSO takes the longest CPU time, and it is not stable with a fixed TSP instance and number of vehicles. In MOPSO, each particle has a private Pareto front set, and the number of solutions in the private sets may be of great difference among different runs. Note that, for kroB150, the CPU time used by the proposed MOACS is almost unchanged with a different number of vehicles, because the complexity of the proposed approach is mainly determined by the number of tasks.

IGDs [22] were also computed for a more accurate analysis of the three algorithms. IGD is an indicator that shows how far is the real Pareto front from the approximate Pareto front set. Let $\hat{\mathcal{P}}$ be a set of uniformly distributed points in the objective space along real Pareto front, and $\mathcal{P}$ be an approximate set, the IGD from $\hat{\mathcal{P}}$ to $\mathcal{P}$ is defined as

$$D(\hat{\mathcal{P}}, \mathcal{P}) = \frac{\sum_{v \in \hat{\mathcal{P}}} d(v, \mathcal{P})}{|\hat{\mathcal{P}}|} \tag{19}$$

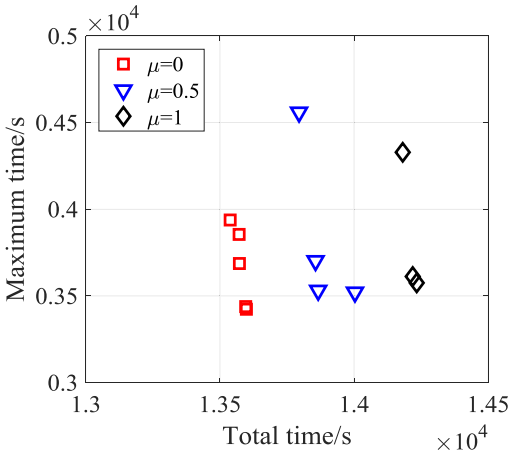


FIGURE 5. Non-dominated fronts obtained by the proposed MOACS on instance named as kroB150 with different kinds of cost definitions.

where v is an element in the real Pareto front, $d(v, \mathcal{P})$ is the minimum Euclidean distance between v and points in $\mathcal{P}$. If $\hat{\mathcal{P}}$ is large enough to represent real Pareto front well, $D(\hat{\mathcal{P}}, \mathcal{P})$ can measure both the diversity and convergence of $\mathcal{P}$.

In order to compute the IGD values, $\hat{\mathcal{P}}$ was set to the set that contains all non-dominated solutions found by all the runs of all the three algorithms. For a specified TSP instance, the number of iterations was set to 2000, and 20 independent runs were performed for each of the three algorithms. Among all the solutions obtained by the three algorithms, the non-dominated ones were extracted as $\hat{\mathcal{P}}$.

Fig. 4 depicts the IGD values obtained by the three algorithms along with iterations, in which IGD values are the average of 20 runs. As shown in Fig. 4, the IGD values of the proposed MOACS are always less than NSGA-II and MOPSO at different iterations, which shows that the proposed MOACS is more efficient in reducing IGD values than NSGA-II and MOPSO.

C. COST DEFINITION

The time cost of vehicle i from task j to k is defined as (1), which refers to the sum of the cruising time from the former task to the latter one and executing time of the former task.

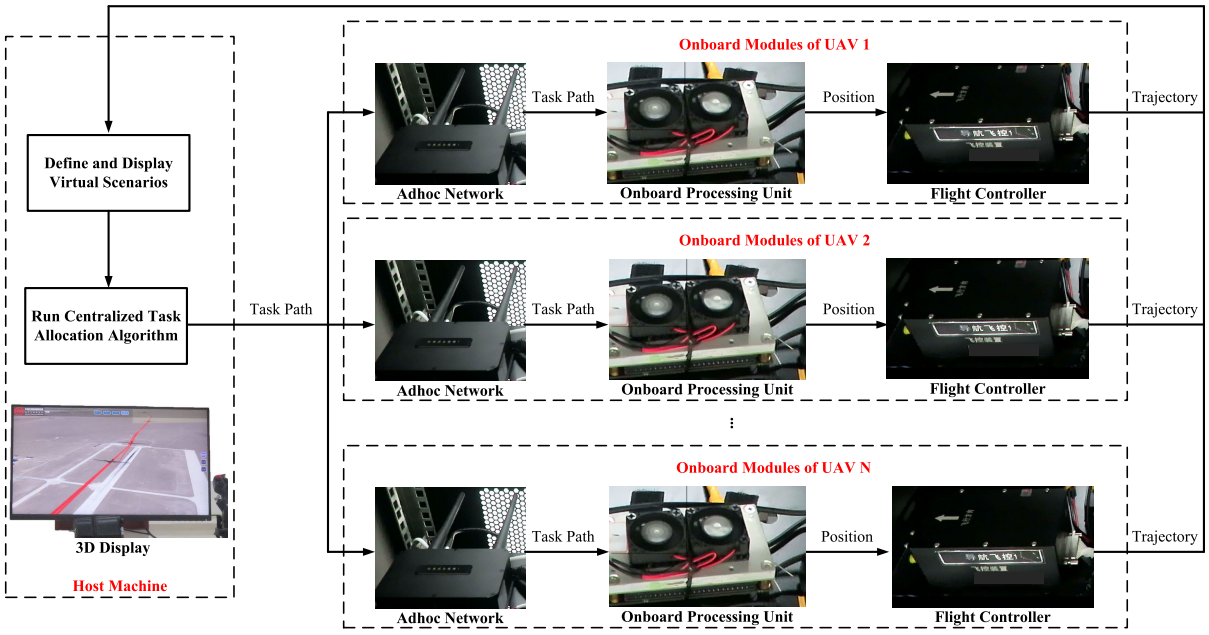


FIGURE 6. Main scheme of the hardware-in-the-loop experimental platform with real Adhoc networks, onboard processing units and flight controllers.

Similarly, given any μ in $[0, 1]$, the time cost can also be defined as

$$c_{jk}^{(i)} = d_{jk}/u_i + (1-\mu)t_{ij} + \mu t_{ik} \tag{20}$$

In this section, three values of μ were considered. $\mu = 0$, which is the same as (1). $\mu = 0.5$, which represents that the cost is defined as the sum of the cruising time and half of the executing time of both the former and latter tasks. $\mu = 1$, which represents that the cost is defined as the sum of the cruising time and executing time of the latter task.

The test scenario is kroB150, the cruising speeds of all vehicles were set to 20 m/s, and the executing time of each task for different vehicles is assumed to be the same, which is uniformly generated between 50 s and 100 s. The none-dominated front sets obtained by the proposed MOACS with different μ are shown in Fig. 5. As shown in the figure, the solutions with $\mu = 0$ dominates those with $\mu = 0.5$, while the solutions with $\mu = 0.5$ dominates those with $\mu = 1$.

In the optimization process of the proposed MOACS, the ants are more willing to select tasks with less time cost from the current task according to (14). Assume that a vehicle is at the former task, then the executing time of this task is necessary despite the value of μ. When $\mu = 0$, the time cost from the current task to the next one being selected is only determined by the cruising time except for the necessary executing time of the former task. However, when $\mu > 0$, the time cost is determined by both the cruising time and the executing time of the next task being selected. Then more cruising time may be needed from the current task to the next one if the executing time of the next task is shorter. Besides, when the executing time of each task performed by different vehicles is assumed to be the same, the executing time of tasks in the total cost is always the same, which equals the total executing time of all tasks. Then the total time cost of vehicles is only dependent on the total cruising time of all vehicles. Therefore, the total time of the vehicles may be bigger due to the more cruising time when $\mu > 0$. The bigger μ is, the cost defined in (20) is more related to the executing time of the latter task, then more time is needed by the vehicles.

V. HARDWARE-IN-THE-LOOP EXPERIMENTS

A. EXPERIMENTAL CONFIGURATION

In order to further investigate the proposed approach, experiments with UAVs were conducted on a hardware-in-the-loop platform. The main scheme of the experimental platform is illustrated in Fig. 6. Real onboard modules are utilized in the platform, which includes Adhoc networks, onboard processing units, and flight controllers, while scenarios, UAVs, and tasks are virtually defined and displayed in a host machine.

As illustrated in Fig. 6, the scenario, UAVs, and tasks are first defined in the host machine. In this experiment, fix-wing UAVs are employed, and tasks are uniformly distributed in the mission area. Then an investigated task allocation algorithm, i.e., MOACS, NSGA-II, or MOPSO, is called to calculate the task paths. Then the task path of each UAV is sent from the host machine to the corresponding UAV. Once the task path is received by the onboard Adhoc network of each UAV, it will be transmitted to and stored by the onboard processing unit. At each step, the target flight position is sent to the flight controller one by one from the onboard processing unit. Once the next flight position is received, the flight controller will calculate the trajectory according to the next flight position, current position, flight speed, and direction. Then the real-time state of each UAV is sent to the host machine from the

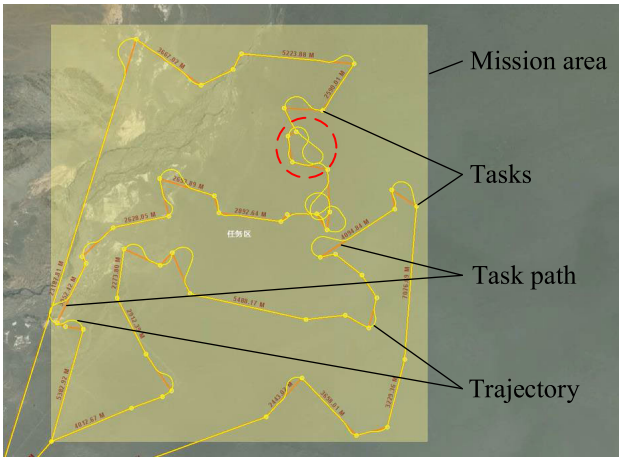


FIGURE 7. The test scenario with 2 UAVs and 50 tasks used in the experiments, mission area, tasks, task paths, and trajectories are illustrated.

flight controller. Finally, the real-time state of the scenario, UAVs, and tasks is updated and displayed in the host machine.

Fig. 7 shows the test scenario in the experiments. The mission area was bounded by $(99.83°E, 40.43°N)$, $(99.83°E, 40.61°N)$, $(100.03°\ E, 40.43°N)$ and $(100.03°E, 40.61°N)$, where N represents northern latitude and E represents east longitude. Fifty tasks were uniformly generated in the mission area. The airport was located at the bottom left corner outside the mission area, which is not illustrated in the figure. Due to hardware limitations, the number of UAVs was set between 2 to 4 in the experiments, and the executing time of tasks was set to 0 since the flight controllers cannot hold the positions of fixed-wing UAVs.

The task paths and trajectories illustrated in Fig. 7 were obtained by the proposed MOACS. As shown in the figure, the trajectories do not always follow the task paths, especially when the next task is close to the current one and the steering angle is big (as shown in the red dashed circle in Fig. 7). This is caused by the limitation of the turning radius, and the trajectories driven by the flight controller will be like "8", which leads to unnecessary cost. Thus in real-world applications, the flight distance may be of great difference even the cost of Euclidean distance is almostly the same.

The turning radius of each UAV is determined by its flight speed, while the flight speed is controlled by its onboard flight controller and varies from time to time. In this experiment, the flight speeds of the UAVs were set to 180 km/h. Due to the properties of the flight controller, the flight speed of a UAV might be more or less than 180 km/h sometimes. As the flight speed was time-varying, the flight time of UAVs was not only determined by their flight distance. Therefore, both the flight time and distance were taken as metrics to evaluate the algorithms.

The result given by each investigated algorithm is a Pareto front set that contains several task paths, but only one task path could be sent to the onboard modules and displayed in the host machine. In most real-world applications, the main advantage of multiple UAVs is efficiency compared with a single one. The time used to finish all tasks is usually more important than the total time used by all UAVs, and it equals the maximum time used by each UAV. Therefore, the task path whose maximum cost is minimum is sent to the onboard modules as the real task path of the UAVs.

TABLE 3. The maximum and total flight distance of the UAVs along the task path of the proposed MOACS, NSGA-II and MOPSO (unit: kilometers).

N_v	MOACS		NSGA-II		MOPSO	
	Max.	Total.	Max.	Total.	Max.	Total.
2	**90.69(2)**	**168.77**	121.19(2)	222.01	119.11(2)	200.51
3	**72.07(1)**	**204.88**	93.78(2)	240.16	85.98(3)	241.74
4	**64.79(3)**	**248.47**	72.08(4)	260.17	76.17(1)	265.47

Here Max. and Total. represent the maximum and total flight distance, and the unit is kilometers. () means the ID of the UAV with maximum flight distance, e.g., 90.69 (2) means that the ID of the UAV with maximum flight distance is 2, and the corresponding flight distance is 90.69 km.

TABLE 4. The maximum and total time used to finish all the 50 tasks along the task path of the proposed MOACS, NSGA-II and MOPSO (unit: minutes).

N_v	MOACS		NSGA-II		MOPSO	
	Max.	Total.	Max.	Total.	Max.	Total.
2	**28.60 (2)**	**54.20**	39.98 (2)	73.58	36.18 (2)	63.37
3	**23.23 (1)**	**64.60**	31.43 (2)	79.65	27.87 (3)	78.72
4	**20.82 (3)**	**76.23**	23.62 (1)	83.00	23.79 (1)	84.36

Here Max. and Total. represent the maximum and total time of the UAVs to finish all the 50 tasks respectively, and the unit is minutes. () represents the ID of the UAV with maximum time, e.g., 28.60 (2) represents that the ID with maximum flight time is 2 and the corresponding flight time is 28.60 min

B. EXPERIMENTAL RESULTS

The maximum and total flight distance to finish all fifty tasks with a different number of UAVs are shown in Table 3. Both the maximum and total flight distances with the proposed MOACS are the smallest. Compared with NSGA-II and MOPSO, the total flight distance used by all UAVs is decreased by up to 23.98% and 15.83% respectively, while the maximum flight distance of each UAV is decreased by up to 25.16% and 23.86%.

The maximum and total time used by a different number of UAVs to finish all fifty tasks are shown in Table 4. Both the maximum and total time used by a different number of UAVs with the proposed MOACS are always the smallest. Compared with NSGA-II and MOPSO, the total flight time used by all UAVs is decreased by up to 26.34% and 17.94% respectively, while the maximum time of each individual UAV is decreased by up to 28.46% and 20.95%.

As shown in Table 3 and 4, as the number of UAVs increases, the maximum flight distance and time decrease, while the total flight distance and time increase. Since all UAVs depart from the depot and return to the depot after finishing all tasks, additional flight distance and time are needed when the number of UAVs increases, thus the total distance and time increase. Note that when the number of UAVs is four and the task allocation algorithm is NSGA-II,

the ID of the UAV with maximum flight distance is 4, while that with the maximum flight time is 1. The difference is caused by the time-varying flight speeds of UAVs. When turning the flight direction, the speed will decrease, and when the trajectory is a long straight line, the speed will increase. Therefore, the ID of the UAV with maximum flight distance may be different from that with maximum time used to finish its tasks.

VI. CONCLUSION

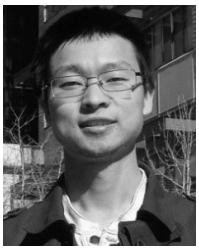
In this paper, a novel MOACS approach has been presented to solve the task allocation problem of multi-robot systems. Since multiple objectives are considered, the standard ACS cannot be applied to solve this problem directly, and a Pareto front-based MOACS algorithm is proposed. In the proposed MOACS, a novel solution construction method is proposed to cope with the multiple salesmen and objectives, which aims to minimize the total cost of all vehicles and balance the cost of each vehicle. Besides, a novel global pheromone update rule is proposed to avoid fast convergence and local optima. Simulation results have shown that the proposed approach has better performance than other well-known approaches within limited iterations over different scenarios. In addition, by computing the IGD values, the proposed approach is shown to be more efficient. Hardware-in-the-loop experiments on multiple UAVs also show that the proposed approach can significantly decrease the flight time and distance of the UAVs. In future work, we are interested in solving the task allocation problem with multiple vehicles performing a complex task cooperatively or a vehicle performing several tasks simultaneously.

REFERENCES

[1] H. Zhang, H. Luo, Z. Wang, Y. Liu, and Y. Liu, "Multi-robot cooperative task allocation with definite path-conflict-free handling," *IEEE Access*, vol. 7, pp. 138495–138511, 2019.
[2] M. Zhao and D. Li, "Collaborative task allocation of heterogeneous multi-unmanned platform based on a hybrid improved contract net algorithm," *IEEE Access*, vol. 9, pp. 78936–78946, 2021.
[3] J. Turner, Q. Meng, G. Schaefer, A. Whitbrook, and A. Soltoggio, "Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system," *IEEE Trans. Cybern.*, vol. 48, no. 9, pp. 2583–2597, Sep. 2018.
[4] Q. Zhong, B. Fang, X. Guo, and C. Zheng, "Task allocation for affective robots based on willingness," *IEEE Access*, vol. 9, pp. 80028–80042, 2021.
[5] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks* (Studies in Computational Intelligence), vol. 604. Springer, 2015, pp. 31–52.
[6] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Studies in Computational Intelligence*, vol. 604, 2015, pp. 31–52.
[7] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.
[8] H.-L. Choi, A. K. Whitten, and H. P. How, "Decentralized task allocation for heterogeneous teams with cooperation constraints," in *Proc. Amer. Control Conf.*, Baltimore, MD, USA, Jun./Jul. 2010, pp. 3057–3062.
[9] W. Zhao, Q. Meng, and P. W. H. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 902–915, Apr. 2016.
[10] A. Whitbrook, Q. Meng, and P. W. H. Chung, "Reliable, distributed scheduling and rescheduling for time-critical, multiagent systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 732–747, Apr. 2018.
[11] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.
[12] S. K. K. Hari, A. Nayak, and S. Rathinam, "An approximation algorithm for a task allocation, sequencing and scheduling problem involving a human-robot team," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2146–2153, Apr. 2020.
[13] C. Nam and D. A. Shell, "Assignment algorithms for modeling resource contention in multirobot task allocation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 889–900, Jul. 2015.
[14] D. Qibo, Y. Jianqiao, and W. Ningfei, "Cooperative task assignment of multiple heterogeneous unmanned aerial vehicles using a modified genetic algorithm with multi-type genes," *Chin. J. Aeronaut.*, vol. 26, no. 5, pp. 1238–1250, Oct. 2013.
[15] X. Kong, Y. Gao, T. Wang, J. Liu, and W. Xu, "Multi-robot task allocation strategy based on particle swarm optimization and greedy algorithm," in *Proc. IEEE 8th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, Chongqing, China, May 2019, pp. 1643–1646.
[16] Y. Chen, D. Yang, and J. Yu, "Multi-UAV task assignment with parameter and time-sensitive uncertainties using modified two-part wolf pack search algorithm," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 6, pp. 2853–2872, Dec. 2018.
[17] Y. Li, S. Zhang, J. Chen, T. Jiang, and F. Ye, "Multi-UAV cooperative mission assignment algorithm based on ACO method," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Big Island, HI, USA, Feb. 2020, pp. 304–308.
[18] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: A multi-objective approach," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2530–2537, Apr. 2020.
[19] Z. Shi, Q. Chen, S. Li, H. Cai, and X. Shen, "Cooperative task allocation for multiple mobile robots based on multi-objective optimization method," in *Proc. 3rd Int. Conf. Comput. Sci. Inf. Technol.*, Jul. 2010, pp. 484–489.
[20] X. Chen, P. Zhang, G. Du, and F. Li, "Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems," *IEEE Access*, vol. 6, pp. 21745–21757, 2018.
[21] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, "Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm," *IEEE Access*, vol. 8, pp. 19306–19318, 2020.
[22] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and ant colony," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1845–1859, Dec. 2013.
[23] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, May 2002.
[24] Y. Shuai, S. Yunfeng, and Z. Kai, "An effective method for solving multiple traveling salesman problem based on NSGA-II," *Syst. Sci. Control Eng.*, vol. 7, no. 2, pp. 108–116, Oct. 2019.
[25] R. Necula, M. Breaban, and M. Raschip, "Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems," in *Proc. IEEE 27th Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2015, pp. 873–880.
[26] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov./Dec. 1995, pp. 1942–1948.

SHENGLI WANG received the B.S. degree in engineering physics from the Department of Engineering Physics, Tsinghua University, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree in engineering physics.

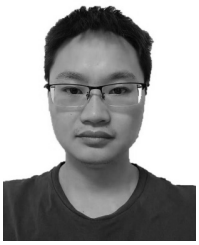
From 2019 to 2021, he was a Research Assistant with the Institute of Electronic Engineering, China Academy of Engineering Physics, Mianyang, China. His research interests include multiple agent systems, swarm intelligence, task allocation, and reinforcement learning.



YOUJIANG LIU received the B.S. and Ph.D. degrees in engineering physics from Tsinghua University, Beijing, China, in 2008 and 2013, respectively.

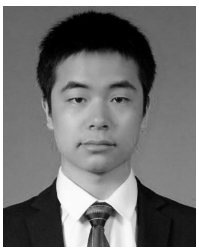
From 2011 to 2012, he was a Visiting Student with the Intelligent RF Radio Laboratory, Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada. From 2013 to 2015, he was a Postdoctoral Researcher with the High-Speed Device Group, Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA, USA. He has been with the Institute of Electronic Engineering, China Academy of Engineering Physics, since 2015, leading an innovative research group. He has authored/coauthored about 40 journals and conference papers. His research interests include signal generation, processing, and circuit design for advanced RF front-ends, such as digital pre-distortion, receive band noise cancellation, time-interleaved analog-to-digital converter calibration, and multiple agent systems. He serves as a Reviewer for the IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON BROADCASTING, and the IEEE MICROWAVE AND WIRELESS COMPONENTS LETTERS.

YONGTAO QIU received the B.S. and Ph.D. degrees in engineering physics from the Department of Engineering Physics, Tsinghua University, Beijing, China, in 2014 and 2019, respectively. He has been with the Institute of Electronic Engineering, China Academy of Engineering Physics, Mianyang, China, since 2016. His research interests include high sampling systems and multiple agent systems.



QI ZHANG received the B.S. degree in engineering physics from the Department of Engineering Physics, Tsinghua University, Beijing, China, in 2012, and the M.S. degree from the Institute of Electronic Engineering, China Academy of Engineering Physics, Mianyang, China, in 2015. His research interests include the area of multiple agent systems and mission planning.

FEIXIANG HUO received the B.S. degree in electronic information engineering and the M.S degree in electromagnetic field and microwave techniques from Sichuan University, Chengdu, China, in 2012 and 2015, respectively. His research interests include large-scale agent system cluster training technology and reinforcement learning algorithms.



YAFAN HUANGFU received the B.S. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2016, and the M.S. degree in communications and information system from the China Academy of Engineering Physics, Sichuan, China, in 2019.

Since 2019, he has been a Research Assistant with the Institute of Electronic Engineering, China Academy of Engineering Physics. His research interests include the full-duplex communication systems and the unmanned aerial vehicle control systems.

CHUN YANG received the B.S. degree from the School of Information Science and Engineering, Shandong University, Qingdao, China, in 1994, and the M.S. and Ph.D. degrees from the Graduate School of the China Academy of Engineering Physics (CAEP), Mianyang, China, in 2001 and 2015, respectively. He is currently with the Institute of Electronic Engineering, CAEP, as a Senior Researcher. His main research interests include communication and information systems.

JIE ZHOU received the B.S. degree from Sichuan University, Chengdu, China, in 1995, and the M.S. degree from the Graduate School of the China Academy of Engineering Physics (CAEP), Mianyang, China, in 2001. He is currently with the Institute of Electronic Engineering, CAEP, as a Senior Researcher. His main research interests include wireless communications and tracking telemetry command and communication.