Received August 4, 2019, accepted September 11, 2019, date of publication September 23, 2019, date of current version October 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942966

Multi-Robot Cooperative Task Allocation With Definite Path-Conflict-Free Handling

HONGGUANG ZHANG, HAN LUO, ZAN WANG, YUHONG LIU, AND YUANAN LIU
School of Electronic Engineering, Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Hongguang Zhang (hongguang-zhang@bupt.edu.cn; hongguang-zhang@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 61876199.

ABSTRACT Modeling and solving multi-robot task allocation with definite path-conflict-free handling is an important research, especially in real working environments. Some of the research lines are unable to obtain definite path-conflict-free solutions for multi-robot task allocations, such as using the penalty-term method in the fitness function to restrict the survival probabilities of the solutions with path conflicts. In some cases, these solutions are only able to satisfy the objective of minimizing task time. We formulate this problem based on grid maps, while focusing on the frequently used cooperative task allocation. In our model, two subtasks of each cooperative task must be executed by two robots, simultaneously. We propose vitality-driven genetic task allocation algorithm (VGTA), which is able to simultaneously minimize task time and realize definite conflict-free path planning. VGTA consists of local operators, such as random mutations, greedy crossovers, and vitality selection. Meanwhile, VGTA includes schedule conflict and path conflict handling strategies. In path conflict handling strategy, we not only consider the common path conflicts in a grid cell, but also focus on the path conflicts between robots when exchanging positions in the adjacent grid cells. Besides, we construct our benchmarks based on real working environments, such as factory, powerhouse, and airport environments. Experimental results indicate that VGTA's search capability and computation cost are satisfactory. Meanwhile, its solutions are able to be really executed.

INDEX TERMS Multi-robot task allocation, cooperative task, schedule conflict, path conflict, unmanned multi-robot swarm.

I. INTRODUCTION

During the past two decades, the innovative technologies emerge endlessly, especially in sensors, chips, and motion controls. These make mobile robots be able to work in the real and complex environments [1], [2], and mobile robots look more like human, such as precisely perceiving, parallel processing, and flexibly moving. The compatibility of mobile robots with real working environments leads to more and more successful applications [3], [4], such as disaster emergency response [5], deep space detection [6], task executions in hazardous environments [7], deep sea research [8], and so on.

From the multi-robot perspective, how to solve multi-robot task allocation and multi-robot path planning simultaneously is one of the important problems in real applications. Generally speaking, according to unknown and known working environments, these researches fall into two categories. (i) One kind of researches is related to unknown working environments (like emergency search and rescue), which are usually characterized by unknown maps, newly added tasks, and intermittent communications. The dynamic and unpredictable nature of unknown working environments is still a serious obstacle to multi-robot cooperative work. (ii) The other kind of researches is related to known working environments (like unmanned multi-robot factory), which are characterized by known maps, known tasks, and all coverage communications. With the commercial deployment of 5G networks, this makes indoor and outdoor coverage communications in cities become possible. Therefore, in known working environments, how multi-robots cooperate with each other and avoid path conflict becomes an important and urgent problem. In this paper, we pursue our research for simultaneously considering both

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

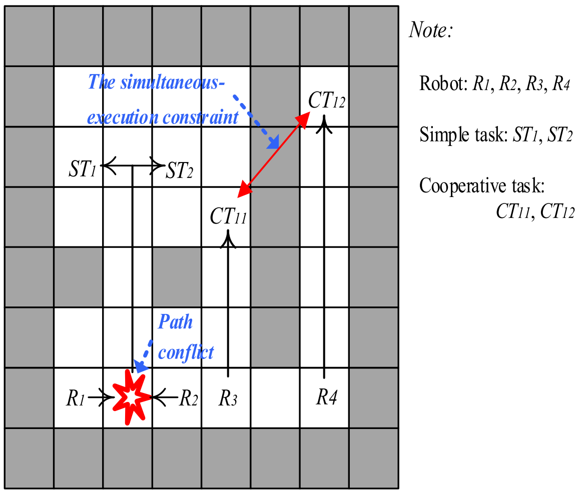


FIGURE 1. The illustration of our research problem.

multi-robot cooperative task allocation and multi-robot definite path-conflict-free planning. Our contributions are as follows.

First, from the research-problem perspective, robot task is classified by robot type (i.e., single-task versus multi-task robots), task type (i.e., single-robot versus multi-robot tasks), and allocation type (i.e., instantaneous allocation versus time-extended allocation) [9]. Fig. 1 provides an illustration of our research problem in this paper, which belongs to single-task time-extended multi-robot allocation problems. In principle, this kind of problems is usually difficult. Moreover, our research problem is characterized by cooperative task allocations [10], [11], and two subtasks of each cooperative task must be executed by two robots, simultaneously. Our difficulties to solve this problem are as follows. (i) Multi-robot task allocation includes cooperative tasks. (ii) Multi-robot mobility leads to path conflicts. We formulate this problem by using grid maps. Note that, our model integrates some of the important factors, such as the schedule conflicts of cooperative tasks, the path conflicts of robots, and the waiting time of cooperative tasks. Besides, we also discuss the limits of our model.

Secondly, from the task-allocation-method perspective, there are some of the classification methods, such as centralized versus decentralized task allocation algorithms, evolutionary computations versus exact solution algorithms, and so on. On the other hand, from the path-planning-method perspective, path conflict handling strategies are classified as definite and indefinite path-conflict-free methods. Definite path-conflict-free methods mean that their solutions do not include any path conflict. Sometimes, the solutions of indefinite path-conflict-free methods (such as using the penalty-term method in the fitness function to restrict the survival probabilities of the solutions with path conflicts) maybe include path conflicts. Our research method is characterized by minimizing task time and providing definite conflict-free path planning, simultaneously. We propose vitality-driven genetic task allocation algorithm (VGTA), which belongs to centralized methods. VGTA is able to minimize task time and provide definite path-conflict-free planning, simultaneously. Besides, we build our benchmarks by using real working environments. Experimental results indicate that VGTA's search capability and computation cost are satisfactory. Meanwhile, we give some of the VGTA's path cases to demonstrate the VGTA's solution is able to be executed in real working environments. Moreover, these VGTA's solutions are definitely path-conflict-free.

The rest of this paper is organized as follows. In Section II, we review related work. In Section III, we present our problem and model. In Section IV, we introduce VGTA. In Section V, we provide experimental results. In Section VI, we give concluding remarks and future work.

II. RELATED WORK

A. MULTI-ROBOT TASK ALLOCATION

Generally, centralized methods of multi-robot task allocation are used in known working environments. Evolutionary computation as a kind of centralized methods is effective to satisfy different requirements, such as task optimizations and path constraints. Zheng and Li describe artificial fish swarm to minimize the task execution time [12], and this allocation approach simulates the fish swarm behaviors, such as swarm, follow, and prey. Li *et al.* establish cloud ant colony algorithm, and cloud model updates pheromone by using expectation, entropy, and hyper-entropy [13]. Chen *et al.* propose multi-robot patrolling problems by using ant colony optimization based memetic algorithm, while simultaneously optimizing the maximum traveled distance and the total traveled distance [14]. Rauniyar and Muhuri study genetic algorithms with adaptive immigrant strategies to address multi-robot coalition formation problem, and this problem mainly formulates the cooperation and coordination between robots and tasks [15]. Meanwhile, in the various fields, there are different formulations of multi-robot task allocation problems, especially for task features and constraint conditions. Guerrero *et al.* introduce a kind of multi-robot task allocation problem's formulation, which focuses on the important task deadlines, such as linear & soft, soft, and hard deadlines [16]. Hussein and Khamis establish a generic market-based approach for heterogeneous robots to fulfill heterogeneous tasks. This formulation considers many of the crucial robot-capability constraints and robot-to-task matching constraints [17].

How to schedule multi-robot in dynamic or unknown working environments autonomously is a difficult problem [18], [19]. Decentralized architectures are widely applied in these challenging environments. Reinforcement learning is an early used architecture in this field [20]. Meanwhile, imitation learning is also used [21], which is characterized by expert's knowledge. Nunes and Gini study an auction algorithm to allocate temporal-constraint tasks, and model the temporal constraints on the tasks as a simple temporal problem [22]. This model is able to maintain consistent schedules, even when adding new tasks. They test their algorithm in the experiments that tasks arrive dynamically. McIntire *et al.* also present iterated multi-robot auction algorithm for precedence-constrained task allocation, in which a batch of tasks that are pairwise unconstrained is selected in each iteration and scheduled by using a modified sequential single-item auction [23]. Nunes *et al.* propose a priority-based iterated sequential single-item auction algorithm for a team of robots to allocate and execute tasks that have temporal and precedence constraints [24]. Meanwhile, they validate their algorithm with three Turtlebot 2 robots and 12 tasks. Chopra *et al.* propose a distributed version of Hungarian method for solving the task assignment problems, and prove that all robots converge to a common optimal assignment under a synchronous implementation [25]. This provides an idea for hybridizing exact-solution-based tools with optimization-based methods. Additionally, multi-robot coalition is an important problem. Luo *et al.* present a distributed auction-based algorithm where the tasks have to be completed within given deadlines [26]. Their problem is to assign tasks to robots, while simultaneously considering the task deadline constraints and the robot's battery life constraints. Note that, when tasks have different task's durations, they prove that their algorithm has an approximation ratio of 2 for this NP-hard problem. Basegio and Bordini propose a decentralized task allocation algorithm, and consider heterogeneous task allocation for heterogeneous agent teams [27]. Moreover, they test their algorithm in various conditions, and discuss the future challenges. The coalition formation algorithm [28] is able to assign the appropriate task set to appropriate robots, and avoid coalition imbalances. To optimize the multi-robot performances continuously, Chen *et al.* present an interesting approach with re-allocating at inter-robot level and re-scheduling at intra-robot level [29].

Recently, with breakthroughs increasing sufficiently, there are many of new application-direction researches in the multi-robot task allocation field. These researches all pay attentions to multi-robot real-task-allocation applications. Xiang and Lee build a new dynamic scheduling system for real manufacturing by combining ant colony intelligence with local agent coordination [30]. Giordani *et al.* propose a decentralized two-level multi agent framework for mobile robot production tasks, which consists of planning level and scheduling level [31]. Jones *et al.* study two time-extended multi-robot coordination methods by using tiered auctions and genetic algorithms [32], respectively. This study investigates intra-path constraints for fire-truck rescue problems [32]. Das *et al.* construct a consensus-based parallel auction and execution framework by means of auction and consensus principles [33], which corresponds to distributed heterogeneous-task-allocation algorithm and healthcare applications. Claes *et al.* tackle the spatial task allocation problem via distributed planning on each robot for warehouse commissioning with the constraint capacity of robots [34]. Note that, they build upon sample-based planning methods, whose performance is independent of the size of the state space in their Monte Carlo tree search. Liu and Kroll [35], [36] propose multi-robot task allocation for inspection problems by combining evolutionary-computation-based algorithm with local operators, which is extended by Jose and Pratihar [37], especially in respect of path collision avoidance. Li *et al.* present hierarchical decision making and trajectory planning method for autonomous farming applications [38], such as harvesting, scouting, and pruning. This method considers cooperative agricultural-robot-task assignment, which includes the cooperative level for formation task assignment and the individual level for agricultural robots' trajectory planning [38]. Overall, these systems or frameworks consider many of real complexity-based and uncertainty-based factors, and this makes them closer to really used application systems.

B. PATH CONFLICT HANDLING RESEARCH

The path planning problem is an interdisciplinary subject between the path planning field and the multi-robot field [39]–[42]. In the multi-robot field, path conflict handling is an intractable problem. In the vast majority of cases, path conflict avoidance directly means avoiding the failures of task executions. Therefore, path conflict avoidance is very crucial to implement multi-robot task allocation. Worth noting that Tsardoulias *et al.* recently provide a new global view of path planning, especially for benchmarks and evaluation metrics [43]. From the multi-robot perspective, most of path-conflict-handling researches are about path conflict free strategies. Spensieri *et al.* propose an iterative and decoupled approach for collision free routing and scheduling, which consists of the sequences for the robot operations with neglecting collisions and the reordered operations to avoid conflicts [44]. This method is applied to an industrial test case, which is adapted from a stud welding station in a car manufacturing line [44]. Wei *et al.* present a novel decentralized approach for multi-robot cooperative pathfinding in dynamic environments, and resolve a set of benchmark deadlock situations by using altruistic coordination [45]. Nam and Shell complete a general model of the multiple-choice assignment problem, and study multiple-choice Hungarian method by using penalization functions [46]. Draganjac *et al.* present a decentralized algorithm for multiple automated guided vehicles performing transportation tasks within industrial and warehousing environments [47]. Besides, this algorithm has been implemented within the robot operating system framework to demonstrate its real usability features [47]. Overall, these lines of recent path-conflict-handling researches are also one of the important requirements in real applications.

This paper considers multi-robot real working applications, which are characterized by known maps and all coverage communications, such as 5G in smart cities, WiFi in unmanned factories. These allow us to establish centralized methods. Generally, centralized methods have the higher algorithm's effectiveness, especially for complex applications. Besides, we provide definite path-conflict-free handling strategy to avoid all of path conflicts in the temporal and spatial dimensions. Generally, this paper presents multi-robot cooperative task allocation with definite path-conflict-free handling strategy.

III. PROBLEM STATEMENT AND FORMULATION

A. STATEMENT AND FORMULATION

First, we explain the relationships among robots, tasks, task allocation, and individual coding as follows.

(i) Robot set R is $\{R_1, R_2, \ldots, R_i, \ldots, R_{NR}\}$, R_i is the ith robot, and $NR= |R|$. All robots are homogeneous, and each robot only executes one task at a time.

(ii) Task set T is $\{T_1, T_2, \ldots, T_j, \ldots, T_{NT}\}$, T_j is the jth task, and $NT = |T|$. There are simple and cooperative tasks in T. Simple task set ST is $\{ST_1, ST_2, \ldots, ST_p, \ldots, ST_{NS}\}$, ST_p is the pth simple task, $NS = |ST|$, and a simple task is executed by a single robot. Cooperative task set CT is $\{CT_{11}, CT_{12}, CT_{21}, CT_{22}, \ldots, CT_{k1}, CT_{k2}, \ldots, CT_{NC1}, CT_{NC2}\}$, and $NC = |CT|/2$. CT_{k1} and CT_{k2} are two subtasks of the kth cooperative task. Note that, CT_{k1} and CT_{k2} must be executed by two robots, simultaneously. For example, two robots simultaneously push one object toward the defined direction. Besides, to simplify the expression of T, ST, and CT, we define that $T = ST \cup CT = \{T_1, T_2, \ldots, T_j, \ldots, T_{NT}\} = \{ST_1, \ldots, ST_p, \ldots, ST_{NS}, CT_{11}, CT_{12}, \ldots, CT_{k1}, CT_{k2}, \ldots, CT_{NC1}, CT_{NC2}\}$, and $NT = NS + 2NC$.

(iii) TA is to express the task allocation matrix in (1). $TA_{i-j} = 1$ means that T_j is allocated to R_i. $TA_{i-j} = 0$ means that T_j is not allocated to R_i. Therefore, (2) is to express the constraint that each task can only be executed by a robot. Besides, (2) and (3) are to express the constraint that all tasks must be executed.

$$TA = \begin{array}{c} \\ R_1 \\ \cdots \\ R_i \\ \cdots \\ R_{NR} \end{array}\begin{array}{c} \begin{array}{ccccc} T_1 & \cdots & T_j & \cdots & T_{NT} \end{array} \\ \begin{bmatrix} TA_{1-1} & \cdots & TA_{1-j} & \cdots & TA_{1-NT} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ TA_{i-1} & \cdots & TA_{i-j} & \cdots & TA_{i-NT} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ TA_{NR-1} & \cdots & TA_{NR-j} & \cdots & TA_{NR-NT} \end{bmatrix} \end{array} \tag{1}$$

$$\sum_{i=1}^{NR} TA_{i-j} = 1 \forall\, 1 \le j \le NT \tag{2}$$

$$\sum_{j=1}^{NT}\sum_{i=1}^{NR} TA_{i-j} = NT \tag{3}$$

(iv) To express the task execution sequence of each robot, we define $CSol = \{CSol_1, CSol_2, \ldots, CSol_i, \ldots, CSol_{NR}\}$ as individual coding in compared algorithms and VGTA. $CSol_i$ is the coding of R_i. We also define $CSol_i = \{CSol_{i-1}, CSol_{i-2}, \ldots, CSol_{i-j}, \ldots, CSol_{i-NCSi}\}$, $CSol_{i-j}$ is the jth task in the task queue of R_i, and $NCSi = |CSol_i|$. Note that, $CSol_{i-j} \in T$. Fig. 2 gives a coding example, and

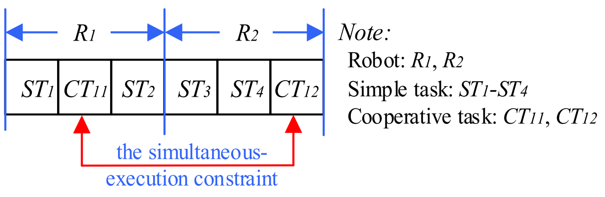

R1
R2
ST1
CT11
ST2
ST3
ST4
CT12
the simultaneous-execution constraint
Note:
Robot: R1, R2
Simple task: ST1-ST4
Cooperative task: CT11, CT12

FIGURE 2. A coding example.

explanations are as follows. R_1 executes ST_1, CT_{11}, and ST_2 in sequence. R_2 executes ST_3, ST_4, and CT_{12} in sequence. Besides, two subtasksCT_{11} and CT_{12} of the cooperative task must be executed by R_1 and R_2, simultaneously.

Secondly, we define the grid map, the initial trajectories, and executable solutions as follows.

(i) Grid map M is to describe the position information of real working environments, such as cells, obstacles, paths, and so on. C is the grid cell set, and $C(x, y)$ is the grid cell at (x, y). The path matrix from the initial positions of robots to the positions of tasks is defined as CS_RT in (4), as shown at the bottom of the next page. CS_RT_{i-j} is the shortest-path-covered cells from the initial position of R_i to the position of T_j, and this kind of the shortest path is obtained by using breadth-first search algorithm. Besides, the path matrix between two tasks is defined as CS_TT in (5), as shown at the bottom of the next page. CS_TT_{j-l} is the shortest-path-covered cells from the position of T_j to the position of T_l, which is also obtained by using breadth-first search algorithm.

(ii) $ITra = \{ITra_1, ITra_2, \ldots, ITra_i, \ldots, ITra_{NR}\}$ is the initial trajectory set, and $ITra_i$ is the initial trajectory of R_i, which is a grid cell subset from C. According to $CSol_i$, CS_RT, and CS_TT, $ITra_i$ is obtained by using (6). For example, $CSol_3 = ST_1, ST_2$. Then, $ITra_3$ is $CS\_RT_{3-1} \cup CS\_TT_{1-2}$. CS_RT_{3-1} is the cell set from the initial position of R_3 to the position of ST_1, and CS_TT_{1-2} is the cell set from the position of ST_1 to the position of ST_2.

$$ITra_i = CS\_RT_{i-CSol_{i-1}} \cup \bigcup_{k=2}^{NCSi} CS\_TT_{CSol_{i-(k-1)}-CSol_{i-k}} \tag{6}$$

(iii) $SolTraWait$ is the executable solution set for all robots in (7). $SolTra_i$ is the executable path cell set for R_i, and c_{i-g} is the gth cell passing by R_i. $SolWait_i$ is the waiting time set for R_i, and w_{i-g} is the waiting time of R_i in c_{i-g}. If R_i does not need waiting in c_{i-g}, then $w_{i-g} = 0$. $NSTW_i$ is the number of cells, which is covered by the path of R_i. According to $CSol$, $ITra$, and T, $SolTraWait$ is obtained by using $RobotRunInMap()$ in Algorithm 7. Note that, $SolTraWait$ is able to satisfy the simultaneous-execution constraint of each cooperative task, and does not include path conflicts.

$$SolTraWait = \begin{bmatrix} SolTra_1 & \cdots & SolTra_i & \cdots & SolTra_{NR} \\ SolWait_1 & \cdots & SolWait_i & \cdots & SolWait_{NR} \end{bmatrix} = RobotRunInMap(CSol, ITra, T) \tag{7}$$

$$\begin{bmatrix} SolTra_i \\ SolWait_i \end{bmatrix} = \begin{bmatrix} c_{i-1} & \cdots & c_{i-g} & \cdots & c_{i-NSTWi} \\ w_{i-1} & \cdots & w_{i-g} & \cdots & w_{i-NSTWi} \end{bmatrix} \tag{8}$$

$$NSTWi = |SolTra_i| = |SolWait_i| \tag{9}$$

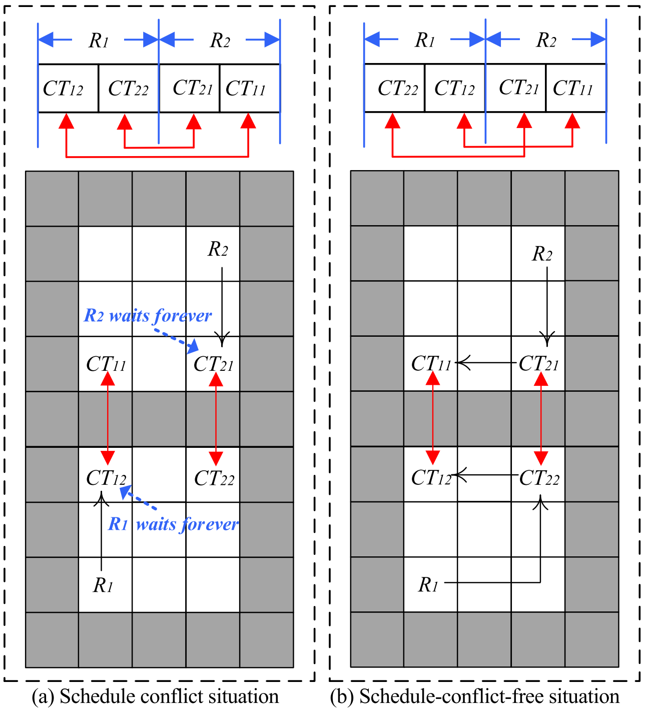
Thirdly, cooperative task and path constraints are the important consideration of our problems. Therefore, we formulate these constraints as follows.

(i) *CSol* is a feasible solution, which does not include any schedule conflict for cooperative tasks. Explanations about schedule conflicts are as follows. Two subtasks of each cooperative task must be executed by two robots, simultaneously. Therefore, there are some of the infeasible solutions with schedule conflicts, as shown in Fig. 3 (a). This kind of schedule conflicts means the individual coding cannot be executed, because two robots wait for each other forever (like R_1 and R_2 in Fig. 3 (a)). We give our schedule conflict handling strategy in Algorithm 5. By using our schedule conflict handling strategy, we can obtain the feasible solution of Fig. 3 (b) from the infeasible solution of Fig. 3 (a).

(ii) In our model, the cell's length is equal to its width, and all robots have the same speed. Then, the time that one robot pass by a cell is able to be defined as one *Unit Time*, and each waiting time w_{i-g} is multiples of *Unit Time*. To simplify the expression of *Unit Time*, *Unit Time* is a dimensionless quantity with no unit. Based on the above assumption, we assume that R_m and R_n execute two subtasks CT_{k1} and CT_{k2} of the kth cooperative task at the c_{m-a} cell and the c_{n-b} cell, respectively. Then, the simultaneous-execution constraint of the kth cooperative task is given in (10). a and b are the index of the ath cell in the R_m path and the index of the bth cell in the R_n path, respectively.

$$a + \sum_{g=1}^{a} w_{m-g} = b + \sum_{g=1}^{b} w_{n-g} \quad \forall 1 \le k \le NC \tag{10}$$

(iii) According to $SolWait_i$ and $SolTra_i$, we are able to obtain the path cell of R_i in any time, as shown in (8). We assume that the current time is *CTime*. Due to ignoring the unit of *Unit Time*, *CTime* is an integer. We define that



R1
R2
CT12
CT22
CT21
CT11
R2 waits forever
R1 waits forever
(a) Schedule conflict situation
(b) Schedule-conflict-free situation

FIGURE 3. Schedule conflict for cooperative tasks.

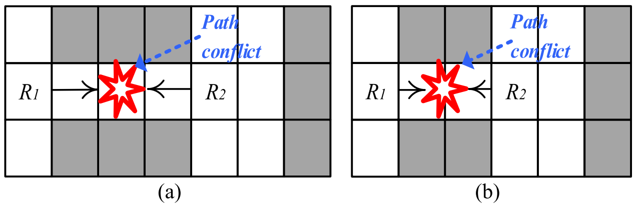


FIGURE 4. Path conflict situations. (a) Path conflict situation I corresponds to (11). (b) Path conflict situation II corresponds to (12).

RCell() returns the path cell of a robot in a specific time. For example, *RCell*(*CTime*, $SolWait_i$, $SolTra_i$) returns the path cell of R_i in *CTime*. We give the constraints that there are no path conflicts in (11) and (12) for two kinds of path conflicts. (11) and (12) correspond to two kinds of path conflicts in Fig. 4 (a) and (b), respectively. Fig. 4 (a) represents path

$$CS\_RT = \begin{array}{c|ccccc|} & T_1 & \cdots & T_j & \cdots & T_{NT} \\ R_1 & CS\_RT_{1-1} & \cdots & CS\_RT_{1-j} & \cdots & CS\_RT_{1-NT} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ R_i & CS\_RT_{i-1} & \cdots & CS\_RT_{i-j} & \cdots & CS\_RT_{i-NT} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ R_{NR} & CS\_RT_{NR-1} & \cdots & CS\_RT_{NR-j} & \cdots & CS\_RT_{NR-NT} \end{array} \tag{4}$$

$$CS\_TT = \begin{array}{c|ccccc|} & T_1 & \cdots & T_l & \cdots & T_{NT} \\ T_1 & CS\_TT_{1-1} & \cdots & CS\_TT_{1-l} & \cdots & CS\_TT_{1-NT} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ T_j & CS\_TT_{j-1} & \cdots & CS\_TT_{j-l} & \cdots & CS\_TT_{j-NT} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ T_{NT} & CS\_TT_{NT-1} & \cdots & CS\_TT_{NT-l} & \cdots & CS\_TT_{NT-NT} \end{array} \tag{5}$$

conflict situation I, when two robots occupy the same cell at the same time. Fig. 4 (b) expresses path conflict situation II, when two robots exchange positions in two adjacent cells. We give path conflict situation II in this paper. Moreover, this kind of path conflict situation is rarely mentioned in other researches, as we known.

$$RCell(CTime, SolWait_i, SolTra_i) \neq RCell(CTime, SolWait_j, SolTra_j)\ \forall 1 \leq i \neq j \leq NRCTime \in \mathrm{Z}+ \tag{11}$$

$$[RCell(CTime, SolWait_i, SolTra_i) == RCell(CTime+1, SolWait_j, SolTra_j)] \cap [RCell(CTime+1, SolWait_i, SolTra_i) == RCell(CTime, SolWait_j, SolTra_j)] = False\ \forall 1 \leq i \neq j \leq NRCTime \in \mathrm{Z}+ \tag{12}$$

Lastly, the task time set ET is $\{ET_1, ET_2, \ldots, ET_i, \ldots, ET_{NR}\}$, and ET_i is the task time for R_i in (13). The fitness function of our model is given in (14), and $max()$ returns the maximum value of a set. Generally, our fitness function is to minimize the maximal value for all robot's task time, while simultaneously considering schedule conflict and path conflict constraints.

$$ET_i = NSTWi + \sum_{g=1}^{NSTWi} w_{i-g} \quad \forall 1 \leq i \leq NR \tag{13}$$

$$minimize(max(\{ET_1, ET_2, \ldots, ET_j, \ldots ET_{NR}\})) \tag{14}$$

B. DISCUSSION

In our model, simple tasks and two subtasks of each cooperative task are the basic tasks, which cannot be further decomposed. Besides, one cooperative task includes two subtasks. In principle, our model is also suitable for more complicated cooperative tasks, such as cooperative tasks with 3 subtasks, 4 subtasks, and so on. We can modify our model by replacing the current definition of cooperative tasks with the new definition of cooperative tasks. We can also select different sets of simple and cooperative tasks to express various task allocation situations. In some sense, the usability of our model is acceptable, especially for multi-robot cooperative task allocation.

There are some limits of our model as follows. (i) As shown in (13), the task time only includes the robot-passing-path time and the waiting time. To compare different task allocation algorithms objectively, we assume that the execution time of each task is 0. That is to say, we ignore the execution time of each task in our model. This is reasonable to compare with various algorithms. However, the execution time of each task should be defined in real applications. (ii) In real environments, there are many kinds of path conflicts, such as robot-arm conflicts. As shown in (11) and (12), we simplify the content of path conflicts by using the following assumptions. First, all robots are homogeneous. Secondly, the cell's length is equal to its width. Lastly, the robot's length and width are less than or equal to the cell's length. From a highlighting-research-objective perspective, these above assumptions are reasonable. However, these assumptions may be not enough in some cases. We believe that grid maps are still a good scheme in real environments, because it is able to describe the complex conflict situations, especially when the cell's shape is far less than the robot's shape. Therefore, we must define the path-conflict-condition matrix for all robots in some cases, and modify the path conflict constraints in (11) and (12) by using the path-conflict-condition matrix.

IV. VITALITY-DRIVEN GENETIC TASK ALLOCATION ALGORITHM

A. MOTIVATION

Our motivation is to obtain definite path-conflict-free solutions for multi-robot cooperative task allocation, thereby guaranteeing the VGTA's effectiveness in real environments.

B. FRAMEWORK

In Algorithm 1, VGTA realizes local search by using random mutation, greedy crossover, and vitality selection [48]. Besides, VGTA uses schedule conflict and path conflict handling strategies to ensure that the VGTA's solutions are definitely schedule-conflict-free and path-conflict-free.

Algorithm 1 VGTA

```
Input: population size $PS$, coding length $CL$(i.e., $CL = NT$), mutation probability $p_m$, the number of executing mutation $NEM$, the number of executing crossover $NEC$, the initial vitality value $V_{ini}$, the maximal vitality limit $V_{max}$, the minimal vitality limit $V_{min}$.
Output: the best solution.
Step 1: Generate an initial population by using Algorithm 2, and use Algorithm 5 to realize schedule conflict handling of these initial individuals.
Step 2: Evaluate the population by using (14). Meanwhile, realize path conflict handling by using Algorithm 6, and obtain waiting time by using Algorithm 7.
Step 3: Execute random mutation process by using Algorithm 3, and obtain the population $MP\text{-}II$.
Step 4: Execute greedy crossover process by using Algorithm 4, and obtain the population $CP\text{-}II$. Note that, in Algorithm 4, we evaluate each individual, and realize schedule conflict and path conflict handling.
Step 5: Execute vitality selection [48] for $CP\text{-}II$.
Step 6: Check whether termination condition is satisfied or not. If it is not satisfied, go to Step 3.
Step 7: The best individual represents the final solution.
```

1) INITIAL POPULATION GENERATION METHOD

A coding example, which corresponds to an individual in the population of VGTA, is given in Fig. 2. We use greedy strategy to generate the initial population as follows.

2) LOCAL OPERATORS

Random mutation process is given in Algorithm 3. Besides, Fig. 5 introduces each kind of random mutations, and their

```
Algorithm 2 Generate the Initial Population
  Step 1: We randomly allocate cooperative tasks to robots,
          generate 5 cooperative task sequences, and use the
          best one as the initial cooperative task sequence.
  Step 2: Each randomly-selected simple task is inserted
          into the best position of the initial cooperative task
          sequence, and the best position is able to
          realize minimizing the task time of the
          newly-inserted-task sequence. Then, obtain an
          initial individual.
  Step 3: Repeat Step 1 and Step 2, until we generate all
          individuals.
```

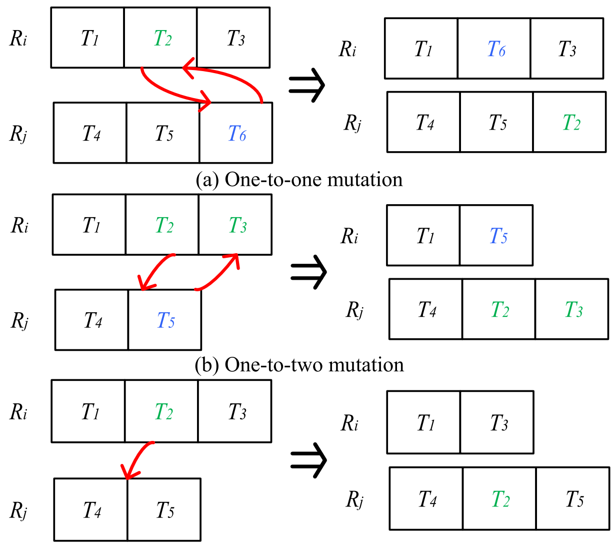


FIGURE 5. The procedures of random mutations. Note that, the mutated tasks, R_i and R_j are all selected randomly.

functions are to protect population diversity. One-to-one and one-to-two mutations realize the task exchange between two randomly-selected robots. Migration mutation is to randomly migrate a task from a randomly-selected robot to another robot.

```
Algorithm 3 Random Mutation Process
  Obtain $PS \times (1-p_m)$ best individuals as the best population
  BP. Meanwhile, use the rest of individuals as mutation pop-
  ulation MP-I. The mutation process of MP-I is as follows.
  Step 1: Randomly choose a kind of mutation operators,
          randomly choose an individual from MP-I,
          and execute NEM mutations.
  Step 2: After completing the mutations of all individuals in
          MP-I, obtain MP-II.
```

Greedy crossover process is given in Algorithm 4. Besides, Fig. 6 gives each kind of greedy crossovers, and their functions are to reproduce new offspring. Greedy crossovers only consider the task time of the local task sequence, such as et_1, et_2, and so on. Note that, these crossovers do not

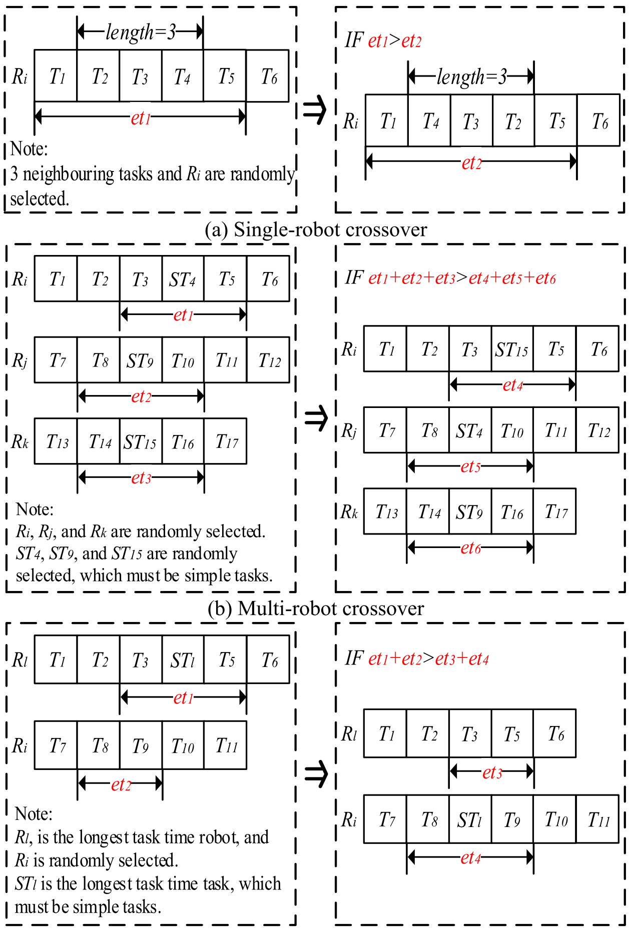


FIGURE 6. The procedures of greedy crossovers. Note that, *eti* expresses the task time of the local task sequence.

compute the execution time of the total task sequence to reduce computation costs, and do not consider path and schedule conflicts, because of its greedy strategy. Their procedures are as follows. (i) In Fig. 6 (a), randomly select an individual, a robot task sequence from this individual, and 3 neighboring tasks from this robot task sequence. Rearrange these 3 tasks. If finding a better sequence, replace the old sequence and stop. Otherwise, repeat the above process, until we complete all arrangement schemes. (ii) In Fig. 6 (b), randomly select an individual, its 3 robot task sequences, and 1 simple task from each robot task sequence. Rearrange these 3 tasks. If finding a better sequence, replace the old sequence and stop. Otherwise, repeat the above process, until we complete all arrangement schemes. (iii) In Fig. 6 (c), randomly select an individual, its longest-time-robot task sequence, and a simple task ST_l from this robot task sequence. Randomly select another robot R_i. Then, insert ST_l into the best position of the robot R_i task sequence.

We use vitality selection (see Algorithm 1 in [48]) as the selection scheme. Besides, theoretical analysis of vitality selection is introduced by us in Section 2.5 of [48], such as the capacities of escaping from local optimum and its

Algorithm 4 Greedy Crossover Process

```
Obtain crossover population CP-I that is the combination of BP and MP-II. The crossover process of CP-I is as follows.
Step 1: Randomly choose a kind of crossover operators,
        randomly choose an individual IND-I from CP-I,
        execute NEC crossovers, and obtain a new
        individual IND-II.
Step 2: Evaluate IND-II. In the process of evaluating
        IND-II, Algorithms 5, 6, and 7 may be used, when
        we find path or schedule conflicts. If IND-II is
        better than IND-I, copy IND-II into the new
        crossover population CP-II. Otherwise, copy
        IND-I into CP-II.
Step 3: After completing the crossover process of all
        individuals in CP-I, obtain CP-II.
```

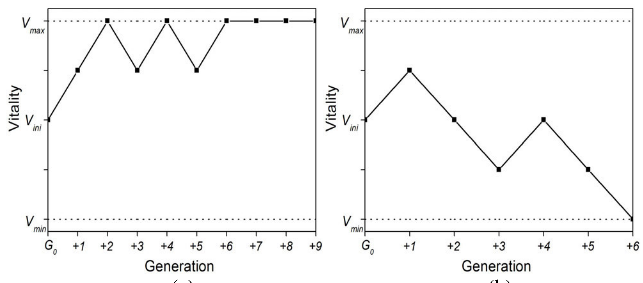


FIGURE 7. Examples of the vitality for different individuals.

behavior features. The vitality of an individual is the life expectancy of this individual in the evolution process. When an individual makes the fitness progress in one generation, we add 1 to the vitality value of this individual. Otherwise, we subtract 1 from the vitality value of this individual. For example, the individual in Fig. 7 (a) that frequently makes the fitness progress is protected from the G_0th generation to the $(G_0 + 9)$th generation. On the contrary, the individual in Fig. 7 (b) that frequently makes no progress is discarded in the $(G_0 + 6)$th generation.

C. CONFLICT HANDLING AND WAITING TIME

We explain schedule conflict and path conflict handling strategies as follows. Meanwhile, these two kinds of handling strategies are related to the waiting times. Therefore, we also explain how to compute waiting times.

1) SCHEDULE CONFLICT HANDLING STRATEGY

There is a schedule conflict in Fig. 3 (a), because two robots wait for each other forever. When schedule conflicts are found in one individual coding, the exchange method is used to adjust the execution order of cooperative tasks. Then, obtain a feasible solution with no schedule conflict, as shown in Fig. 3 (b). Schedule conflict handling strategy is given in Algorithm 5.

Algorithm 5 Schedule Conflict Handling Strategy

```
Step 1: Repeat Step 2 for each individual, until all individuals are checked.
Step 2: Check whether the individual includes schedule conflicts or not. If this individual includes schedule conflicts, repair schedule conflicts by using the exchange method from Fig. 3 (a) to Fig. 3 (b).
```

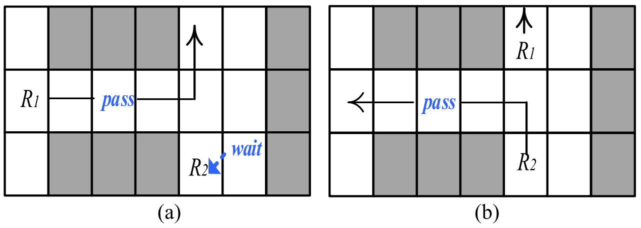


FIGURE 8. An illustration of path conflict handling strategy.

2) PATH CONFLICT HANDLING STRATEGY

We consider two kinds of path conflict situations, as shown in Fig. 4. Our path conflict handling strategy in Algorithm 6 uses the waiting method, as shown in Fig. 8 (a) and (b). When path conflicts are found, our path conflict handling strategy adjusts the shorter-time-robot path, moves this robot to the nearest & feasible grid, and waits for another robot to pass. Note that, this nearest & feasible waiting cell must not be covered by any robot path.

Algorithm 6 Path Conflict Handling Strategy

```
Step 1: Choose the robot with the shorter-time path. This
        robot uses the path-conflict cell as the initial cell,
        and searches the nearest & feasible waiting cell.
        Note that, this nearest & feasible waiting cell must
        not be covered by any robot path. Then, this robot
        moves to the waiting cell, and waits for another
        robot to pass, as shown in Fig. 8(a).
Step 2: After another robot passes, this robot continues to
        finish tasks, as shown in Fig. 8 (b).
```

3) WAITING TIME COMPUTATION METHOD

In our model, there are two kinds of waiting time. One is the waiting time due to path conflict, as shown in Fig. 8 (a). Another is the waiting time due to the simultaneous-execution constraint of two subtasks in each cooperative task, as shown in Fig. 1 and (10). *SolTraWait* is obtained by using Algorithm 7.

D. DISCUSSION

VGTA belongs to centralized methods, which should be running in a central computer of multi-robot task allocation. Therefore, the computation cost of VGTA is not restricted to the low computing capability and finite memory of each robot. When VGTA provides its best solution, the central

TABLE 1. Benchmarks. *NS*, *NC*, and *NR* are the number of simple tasks, cooperative tasks, and robots, respectively. BKS is the best known solution. Besides, we simplify the expression of BKS by using $CSol_i$. For example, $CSol_1$ also be expressed by "R1: 1, 2". This means that R_1 sequentially executes T_1 and T_2.

Test set	No.	Map			Task			BKS
		Environment	Size (cell)	Reference	*NS*	*NC*	*NR*	
	F1	Manually generated	25×15	No	6	2	3	R1:2,1,4,3; R2:7,5,8; R3:9,6,10;
A small scale test set	F2	Manually generated	25×20	No	6	2	3	R1:10,9,4; R2:1,2,6,5; R3:8,7,3;
	F3	Manually generated	18×17	No	6	2	3	R1:1,2,5; R2:10,6,4,7; R3:8,3,9;
	F4	Factory	45×52	[36]	50	5	3	R1:5,27,29,28,54,49,48,47,46,44,16,53,55,13,12,56,40,14,52;R2:4,6,59,50,43,42,45,58,15,17,18,19,20,21,22,23,24,25,26,1,2,3;R3:7,8,9,35,41,39,38,36,37,60,11,51,10,34,33,31,32,57;
	F5	Simulation	30×38	[53]	25	5	3	R1:1,10,9,26,32,33,23,25,21;R2:2,3,12,13,14,15,16,27,24,28,17,34,35;R3:4,5,31,6,7,8,11,20,18,19,22,29,30;
	F6	Community	40×30	[52]	20	5	3	R1:1,25,19,7,24,22,13,23,21;R2:30,15,29,9,10,27,11,12,26;R3:2,3,4,5,6,8,20,18,16,17,28,14;
	F7	Workplace	30×40	[51]	20	5	3	R1:1,30,5,29,23,14,22,16,15;R2:25,24,13,28,27,20,7,8,26;R3:2,3,4,17,19,18,12,11,10,6,9,21;
	F8	Workplace	30×40	[51]	15	5	3	R1:2,1,21,22,23,9,6,7,8,4;R2:16,17,18,19,20;R3:3,5,10,11,15,12,13,24,14,25
	F9	Workplace	60×60	[51]	30	5	3	R1:2,34,17,38,30,36,18,37,28,23,22,24;R2:1,39,33,21,19,20,31,29,32,15,14,16,40,13;R3:3,4,5,6,7,8,9,26,27,10,11,12,25,35;
A large scale test set	F10	Workplace	50×50	[51]	20	5	3	R1:2,8,7,11,29,13,23,30,21,27;R2:1,3,4,9,14,24,15,28,20,26,22;R3:5,6,10,17,18,19,25,16,12;
	F11	Workplace	40×70	[51]	25	5	3	R1:35,33,18,29,10,9,32,31;R2:28,22,34,19,20,16,14,21,13,1,17;R3:2,5,4,7,6,25,8,23,11,24,27,12,26;
	F12	Powerhouse	50×50	[50]	30	5	3	R1:9,12,40,24,37,36,34,26,33,29,27;R2:13,35,23,32,31,4,3,5,6,10,15,14,11; R3:1,2,7,8,28,21,20,19,17,16,39,18,22,25,38,30;
	F13	Airport	30×70	[49]	20	5	3	R1:1,6,28,9,10,3,25,21,4,20;R2:7,23,8,24,11,12,13,14,16,17,27;R3:2,5,29,19,30,15,26,18,22;
	F14	Workplace	50×50	[51]	45	5	4	R1:1,3,54,45,53,55,10,51,52,44,34,18,21;R2:12,40,35,49,15,14,50,13,43,22,30,25,26;R3:9,11,17,16,39,19,48,20,41,42,28,38,29,33,27;R4:7,23,37,4,5,6,8,46,47,24,32,36,31;
	F15	Workplace	50×50	[51]	45	5	5	R1:51,55,53,1,2,3,4,5,38;R2:50,30,44,17,31,23,15,28,16,25;R3:52,33,14,13,9,7,8,10,39,24,12,11,29;R4:48,40,36,35,37,43,6,34;R5:47,18,19,20,21,26,32,27,22,42;
	F16	Workplace	40×70	[51]	45	5	6	R1:15,30,35,1,2,54,52,26,12,31;R2:5,3,49,45,47,51;R3:41,24,11,32,50,9,10,27,23;R4:4,7,25,6,53,8,40,46,44;R5:13,14,21,36,20,55,37,16,43,17;R6:38,39,42,48,29,19,34,18,33,22,28;

computer communicates with each robot to realize task allocation. VGTA is suitable for all-coverage-communication working environments, especially for 5G in smart cities, WiFi in unmanned factories, and so on.

The algorithm framework of VGTA is suitable for multi-robot cooperative task allocation, and explanations are as follows. (i) The coding method is able to describe different task allocation sequences. Besides, the initial population method is able to generate different kinds of individual coding. (ii) Three kinds of random mutations are able to realize the task regroup among different robots. Three kinds of greedy crossovers are able to optimize the local task sequences by using greedy strategy. Vitality selection is effective to make populations escape from the local optima (see Sections 2.5 and 2.6 in [48]). (iii) Generally, we only revise the problem model for the new kinds of multi-robot cooperative task allocation, and seldom need to modify the algorithm framework of VGTA.

In our problem, two subtasks of all cooperative tasks must be executed, simultaneously. This leads to complex temporal and spatial constraints. How to solve complex temporal and spatial constraints is a difficult problem. (i) From a temporal constraint perspective, our schedule conflict handling strategy is able to provide schedule-conflict-free task execution solution. (ii) From a spatial constraint perspective, our path conflict handling strategy uses the waiting method, which is a kind of definite path-conflict-free method. (iii) These mean that VGTA's solutions not only satisfy temporal-based schedule-conflict-free constraints, but also satisfy spatial-based path-conflict-free constraints. From a temporal and spatial constraint perspective, our task allocation solutions are able to be really executed.

There are some of VGTA's limits as follows. (i) All robots are mobile in our problem, and then each robot is restricted to its finite energy. In this paper, we assume that the battery (or fuel) of each robot is far greater than the

TABLE 2. Used parameter settings in this paper.

Items	Parameters
MA	population size *PS*=50, non-overlapping groups *K*=10, mutation probability $p_m = 1$
FSA	population size *PS*=50, crowd factor δ=0.1, times of searching food attempts *TN*=7, visual distance *VD*=5
IPGA	population size *PS*=50, the number of groups=5, breakpoint mutation probability=0.6
GVNS	population size *PS*=50, max shaking times k_{max}=3, max neighborhood structures l_{max}=5
VGTA	population size *PS*=50, mutation probability p_m = 0.9, the initial vitality value V_{ini} =3, the maximal vitality limit V_{max}=6, the minimal vitality limit V_{min}=0. To improve the randomness of executing mutation operators, the number of executing mutation *NEM* = RAND (INT(coding length *CL* /6), INT(coding length *CL* / 8)), RAND(A, B) returns an integer from A to B, and INT(C) returns an integer for a floating number C. With generations increasing, to improve the search effort of crossover operators, the number of executing crossover *NEC* = INT(the current number of computing fitness function / (*PS*×3)).
Termination Condition	We used the number of computing fitness functions as termination condition, which is equal to 1500.
Program and Computer	All of algorithms were programmed in Matlab R2012a. Meanwhile, we executed them on a virtual machine with Windows 7, 4 sub-CPUs of Intel(R) Xeon(R) CPU E50-2680 v2 @ 2.8GHz, and 2GB RAM.

TABLE 3. Wilcoxon test results between each compared algorithm and VGTA.

Algorithm	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
MA	+	+	-	+	+	+	+	+	+	+	+	+	+	+	+	+
FSA	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IPGA	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
GVNS	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+
SUM(+)	4	3	2	4	4	4	4	4	4	4	4	4	4	4	4	4

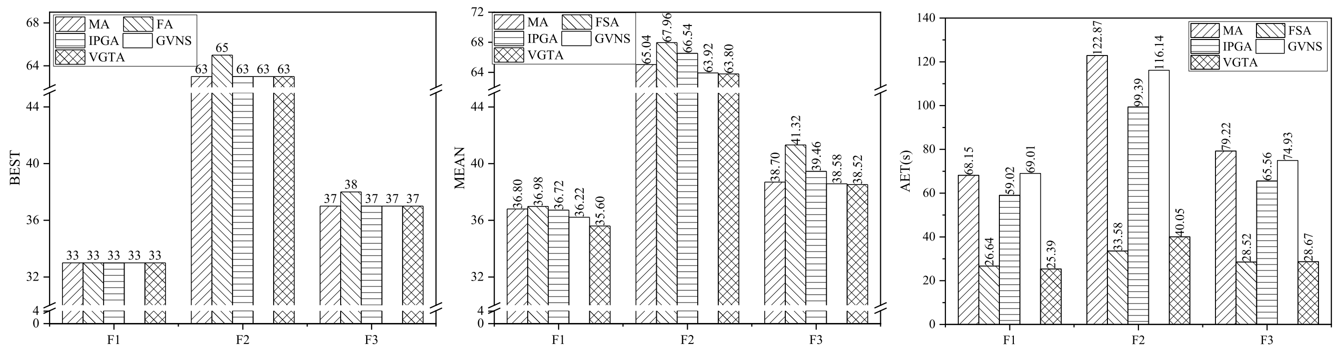


FIGURE 9. BEST, MEAN, and AET results for the small scale test set.

total-task-accomplishment energy. The battery charging (or fuel loading) is necessary in real environments. When VGTA is used in real environments, the simple task of the battery charging (or fuel loading) should be added into a simple task set. Besides, the corresponding constraint should be added into our model, and the corresponding constraint handling strategy should be considered in VGTA. (ii) We assume that the task queue of each robot is long enough for its assigned tasks. However, to avoid the robot-task-queue overflows, the robot-task-queue length should be considered in the hand-shaking protocol between the central computer and robots.

V. EXPERIMENTS AND DISCUSSIONS

A. EXPERIMENTAL DESIGN

1) BENCHMARKS

Our benchmarks were 16 test problems in Table 1, which could be downloaded from https://github.com/hongguang-zhang/MRTA_benchmarks.

In Table 1, a small scale test set was manually generated, and a large scale test set was based on real working environments [36], [49]–[53] with minor modifications (such as adding cooperative tasks).

2) COMPARED ALGORITHMS

We used memetic algorithm (MA) [35], fish swarm algorithm (FSA) [12], IPGA [54], and general variable neighborhood search (GVNS) [55] as compared algorithms. (i) Due to our new benchmarks, there were few compared algorithms for simultaneously considering schedule conflict and path conflict in multi-robot cooperative task allocation, as we known. Therefore, we explained why we used MA, FSA, IPGA, and GVNS as compared algorithms as follows. MA and FSA were homologous algorithms for multi-robot task allocation. Besides, IPGA and GVNS were the general algorithms for multiple traveling salesmen problem,

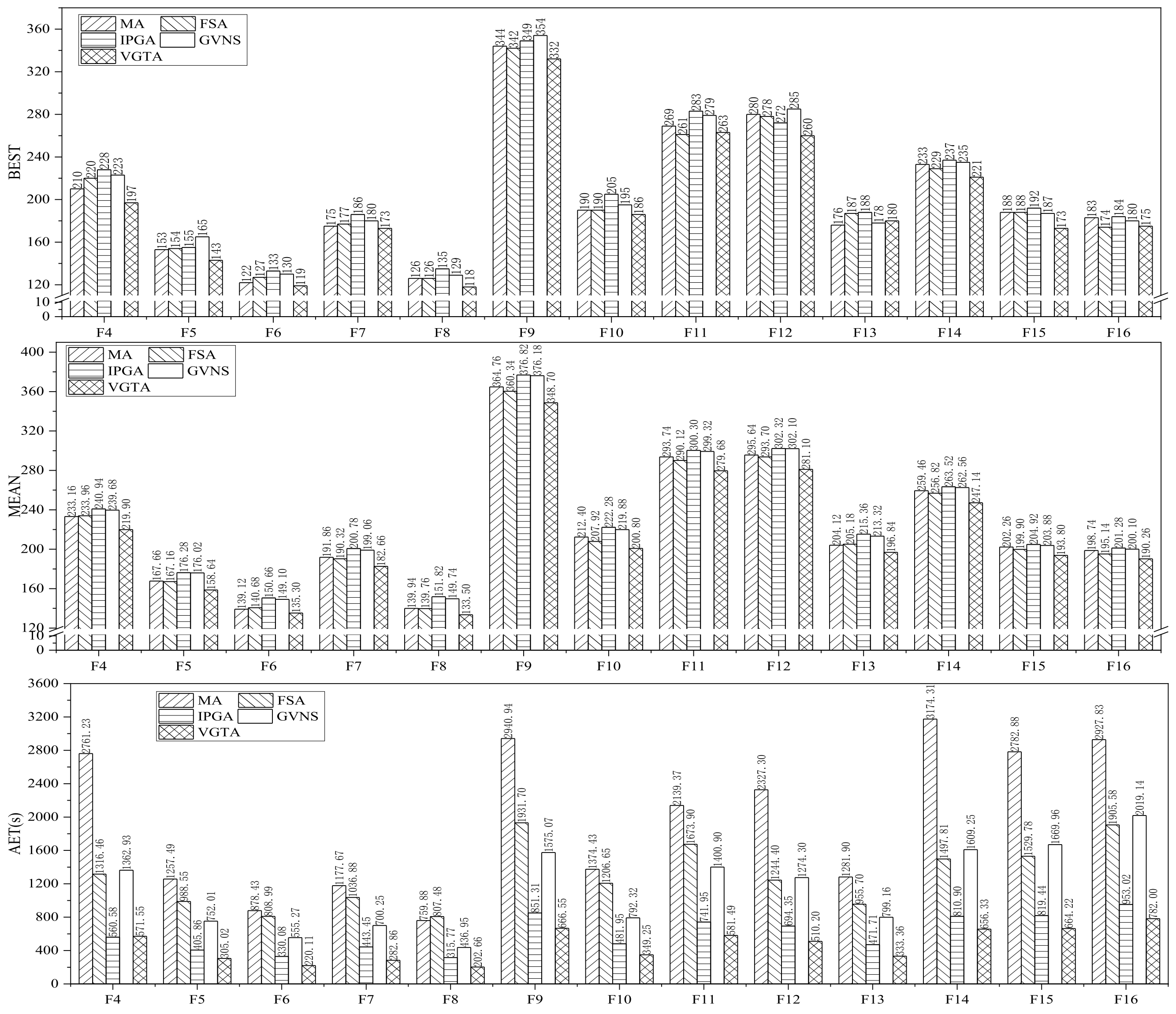


FIGURE 10. **BEST, MEAN, and AET results for the large scale test set.**

which was related to multi-robot task allocation problem. (ii) In addition, compared algorithms also used the same methods of generating initial populations in Section IV. B. 1, schedule conflict handling in Section IV. C, and path conflict handling in Section IV. D. These methods did not change the framework and details of each compared algorithm. We used these methods in each compared algorithm. The reason was to adapt to multi-robot cooperative task allocation in respects of initial population generation, schedule conflict handling, and path conflict handling for each compared algorithm.

3) USED PARAMETER SETTINGS

Used parameter settings were shown in Table 2.

4) EVALUATION CRITERIA

Evaluation criteria were as follows. (i) MEAN was the mean of the best individual fitness for 50 runs. BEST was the best solution for one algorithm in 50 runs. (ii) AET (s) was the average execution time of one algorithm for 50 runs. (iii) The percentage deviation of the best solution *PDbest* in (15) and the percentage deviation of the average solution *PDav* in (16) were used in [56]. BKS was the best known solution. Due to the new benchmarks in this paper, these BKSs in Table 1 were found in the entire experimental process of all algorithms. (iv) The smaller previous evaluation criteria, the better performance.

$$PDbest = \frac{\text{BEST} - \text{BKS}}{\text{BKS}} \times 100 \tag{15}$$

$$PDav = \frac{\text{MEAN} - \text{BKS}}{\text{BKS}} \times 100 \tag{16}$$

B. ALGORITHM'S EFFECTIVENESS AND EFFICIENCY

Firstly, Fig. 9 gave BEST, MEAN, and AET results for the small scale test set. From the algorithm's effectiveness

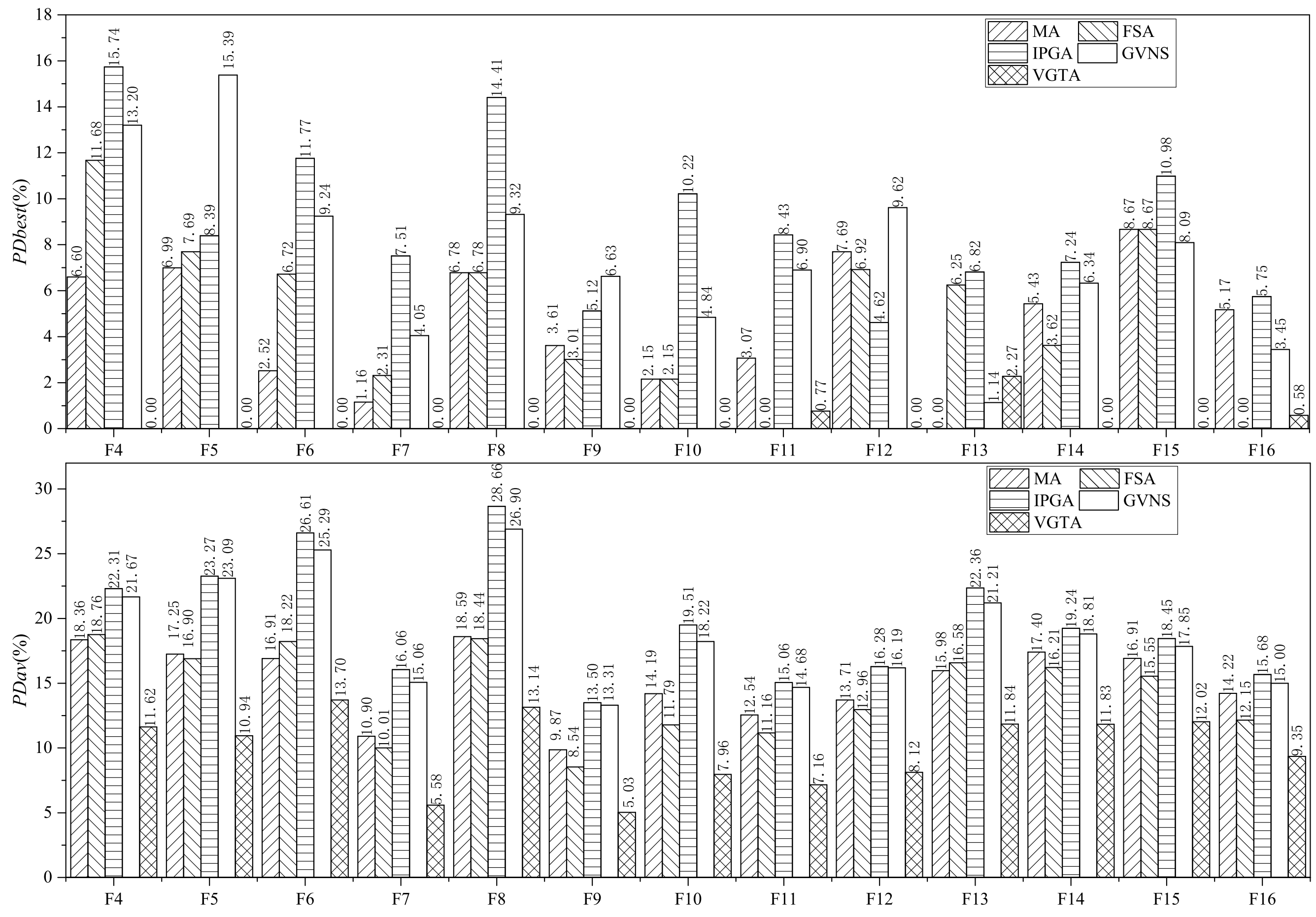


FIGURE 11. ***PDav*** **and** ***PDbest*** **results for the large scale test set.**

Algorithm 7 *RobotRunInMap*()

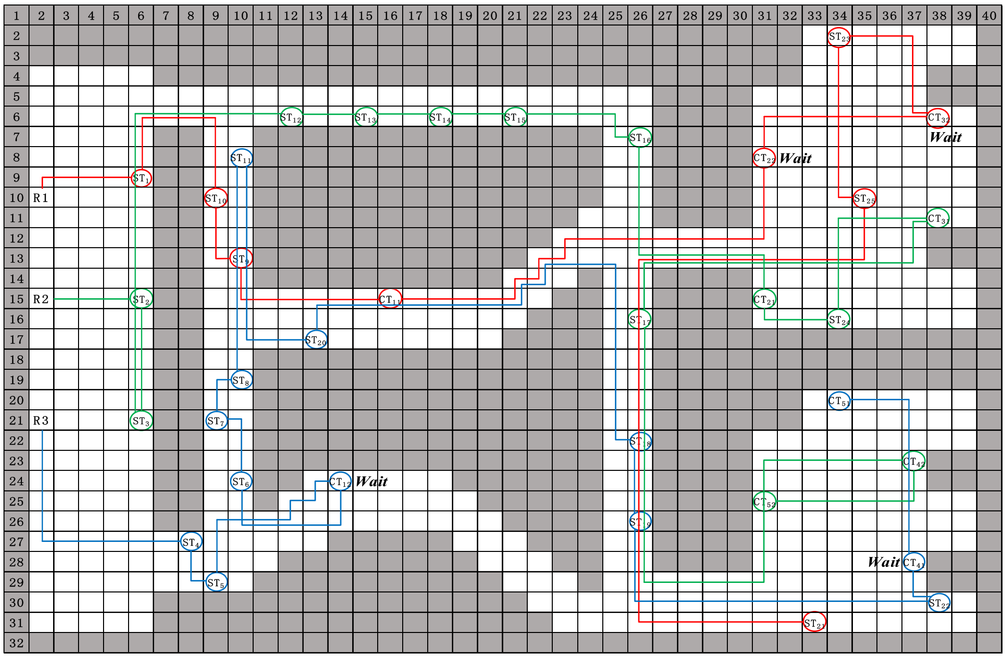
```
Input: $M$, $R$,$CSol$, $ITra$, and $T$
Output: $SolTraWait$
Step 1: Assume that the current time is $CTime$, and
        $CTime = 0$.
Step 2: Repeat Steps 3 and 4, until all tasks are completed.
Step 3: $CTime = CTime + 1$
Step 4: Based on (11) and (12), check any path conflict.
        If one path conflict is found, handle this path
        conflict by using Algorithm 6. Besides, obtain the
        waiting time of the waiting robot (like $R_2$ in Fig. 8).
        Meanwhile, compute the waiting time of each
        cooperative task in the current cell according
        to (10). Then, record the corresponding cells in
        $SolTra_i$, and record the corresponding waiting time
        in $SolWait_i$.
```

perspective, VGTA was the best algorithm for BEST and MEAN results. However, these results of all algorithms were very close. From the algorithm's efficiency perspective, VGTA had some advantages in AETs. However, these advantages were not obvious. Overall, these VGTA's results were not good enough for the small scale test set.

Secondly, Fig. 10 provided BEST, MEAN, and AET results for the large scale test set. The performances of VGTA for the large scale test set were better than the small scale test set, especially for AETs. Generally, VGTA was more suitable for the large scale test set, such as F16 with 45 simple tasks and 5 cooperative tasks.

To further demonstrate the effectiveness of VGTA for the small scale and large scale test sets, we used Wilcoxon test at 0.05 level of significance for the 50-run best-solution-fitness data sets of two algorithms. As non parametric test, Wilcoxon test is widely used to analyze the difference between two data sets. In Wilcoxon test, a plus sign (+) means that the effectiveness of VGTA is significantly better than the compared algorithm, and a minus sign (−) means that the difference between the compared algorithm and VGTA is not significant. In Table 3, the results demonstrated that the effectiveness of VGTA was not very good for the small scale test set. However, from a non-parametric-test perspective, VGTA was significantly better than all compared algorithms for the large scale test set.

From a computation-time and cooperative-task-category perspective, we further discussed the real time performance of VGTA in multi-robot task allocations as follows. Note that, cooperative tasks must need two robots to complete


(a) F5

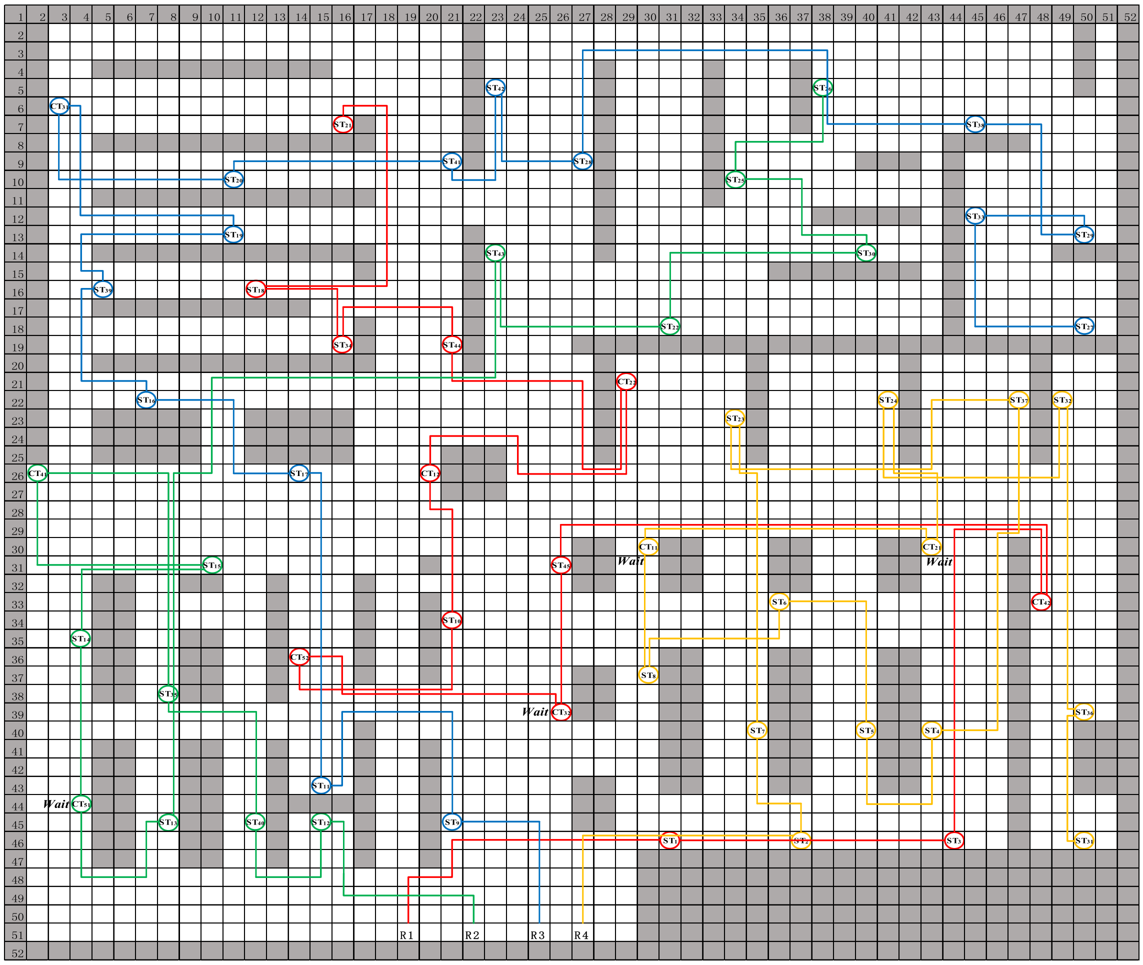
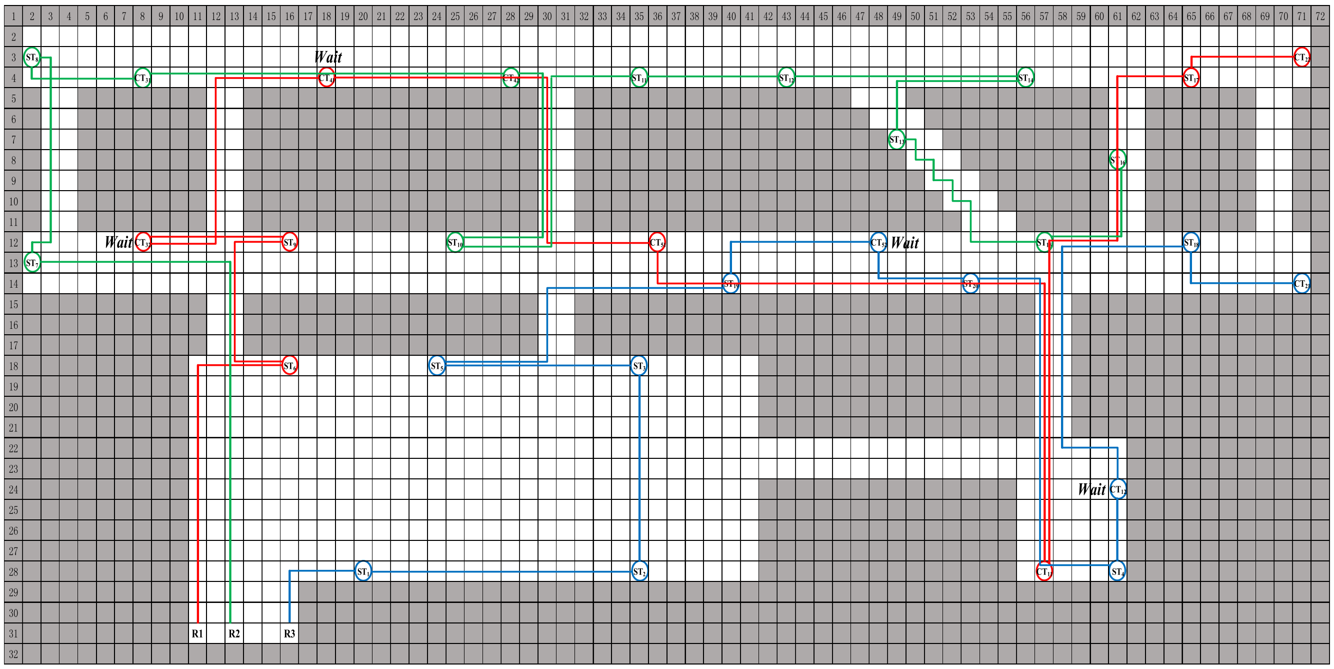
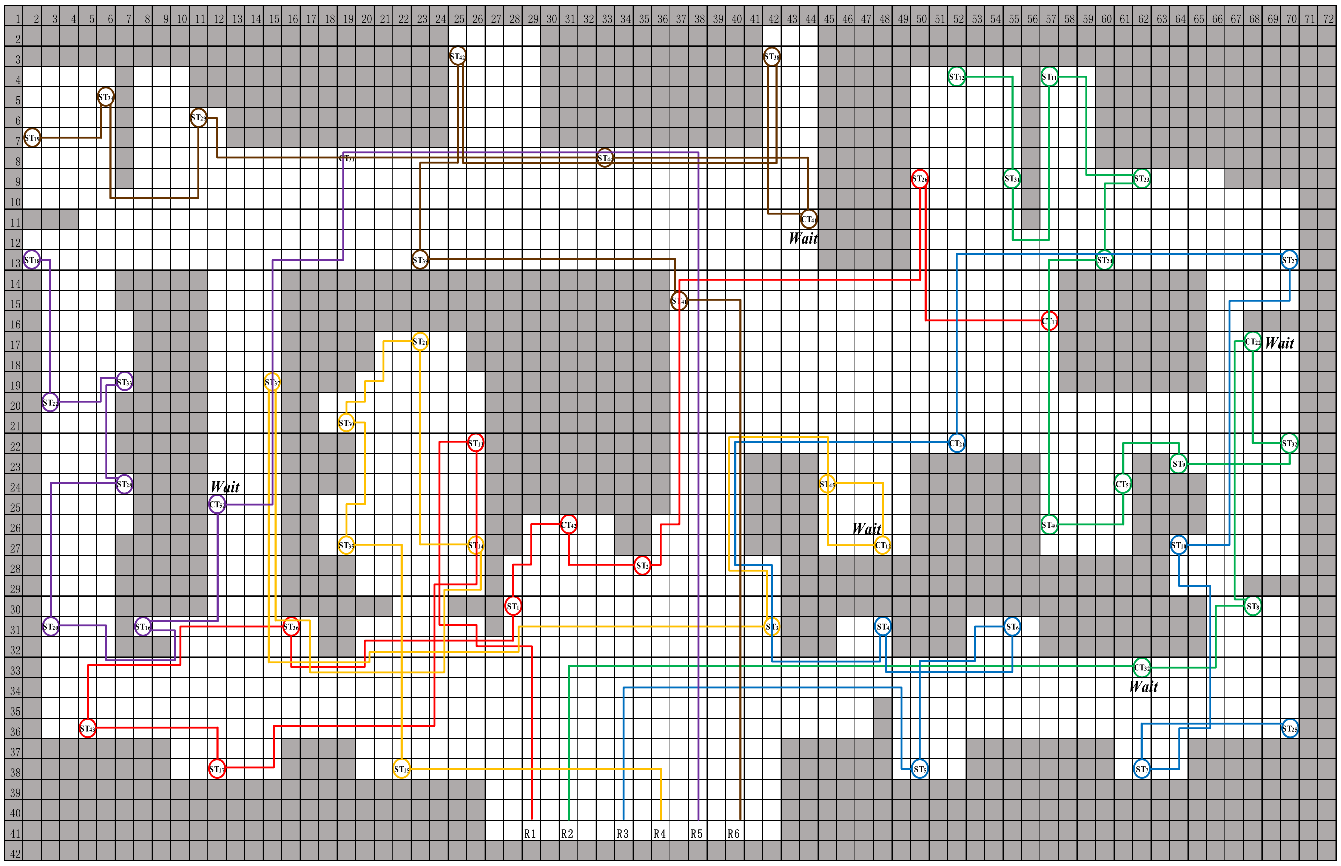

(b) F13

FIGURE 12. Some cases of the visual best-solution paths of VGTA.



(c) F14



(d) F16

FIGURE 12. ***(Continued.)*** **Some cases of the visual best-solution paths of VGTA.**

one cooperative task. In principle, the waiting time among cooperative task robots is very hard to avoid. Moreover, with the number of cooperative tasks increasing, the sum of the waiting time increases obviously. Therefore, in some sense, the real time requirement of multi-robot cooperative task allocation is generally lower than the average level of

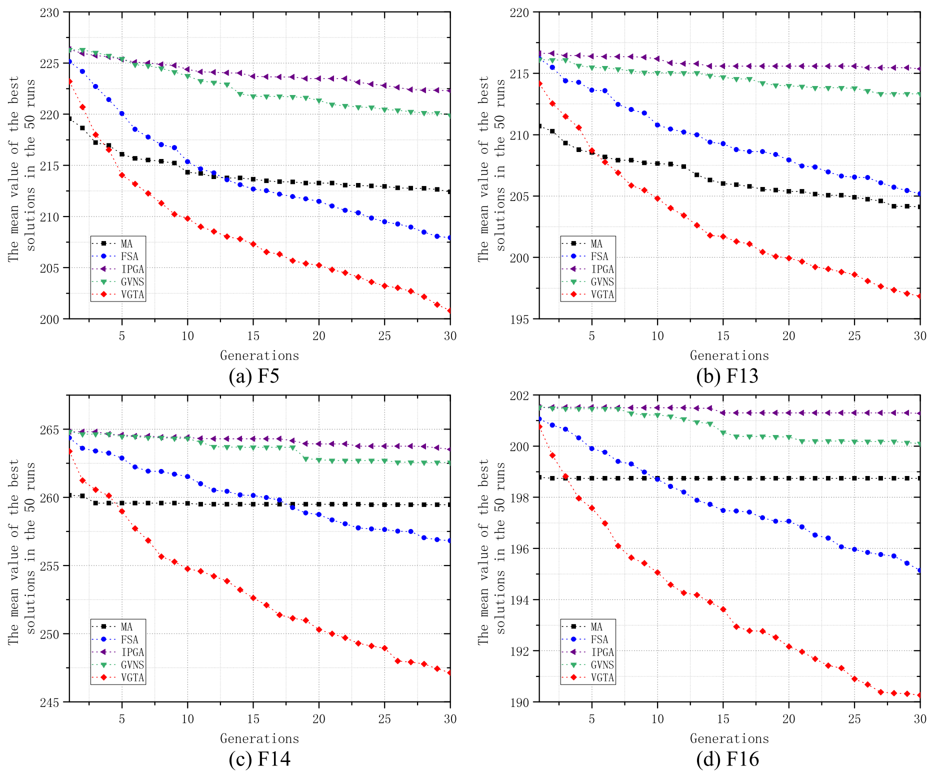


FIGURE 13. The best-solution-mean curves of all algorithms for 50 runs.

multi-robot simple task allocation. In addition, by comparing with other algorithms, AET results of VGTA were satisfactory as a centralized method for the small scale and large scale test sets.

C. APPLICATION'S EFFECTIVENESS AND PATH CASES

Note that, BKS is the best known solution. *PDbest* and *PDav* express the algorithm capability closer to BKS in an optimal-value or mean-value respects, respectively. The smaller *PDbest* and *PDav*, the better algorithm reliability in real applications. Fig. 11 gave *PDbest* and *PDav* results. Except for F11, F13, and F16, VGTA's *PDbest* was the best. VGTA's *PDav* was the best in all cases. Overall, VGTA's reliability was better than others. Meanwhile, engineers also focus on the cost-effectiveness ratio (CER) of an algorithm in real applications. CER is related to computation time and algorithm reliability. The smaller computation time and the higher algorithm reliability mean the better CER. VGTA's AETs in Fig. 10 were the best, except for F4. In summary, by comparing with other algorithms, the application's effectiveness of VGTA was satisfactory from a CER perspective.

To make readers easily understand the physical meaning of the best solution, we gave some of the best solution paths of VGTA in Fig. 12. Note that, these maps were all based on the real working environments, and the complexity of these maps was relatively higher than the existing studies, as we known. In Fig. 12, the path of each robot R_i corresponded to the $SolTra_i$ of the best solution. Besides, we also provided the mean curves of the best solutions of all algorithms in Fig. 13. These convergence curves indicate that VGTA is able to guarantee its convergence speed even in the real-working-environment maps.

VI. CONCLUSION

In multi-robot task allocation, there is relatively little research that focuses on multi-robot cooperative task allocation with definite conflict-free path, as we known. Meanwhile, with problem difficulty (like the number of tasks) increasing, the contradictions between multi-robot cooperative task allocation and definite conflict-free path planning are more and more incisive. In this paper, we formulate the corresponding model, and introduce VGTA that is characterized by providing the executable solution with definite conflict-free path. In addition, we test VGTA by comparing with MA, FSA, IPGA, and GVNS in our real-working-environment benchmarks. These benchmarks are relatively complicated in the last-decade-related research lines, as we known. This work is able to be used in a wide range of indoor or industry multi-robot cooperative working environments, when there are known maps, known tasks, and all-coverage communications (like 5G in the emerging smart cities) with no uncertain-factor influences. Besides, we discuss some of the extensible problems about our model and VGTA, such as the fine-grained path-conflict problem in Section. III. B, the battery-charging (or fuel-loading) problem in Section. IV. E, and the finite length problem of robot task queues in Section. IV. E. Meanwhile, we provide some suggestions for these problems.

In the future, we consider modeling heterogeneous multi-robot cooperative task allocation. Meanwhile, we will study online systems for realizing heterogeneous multi-robot cooperative task allocation in real time.

REFERENCES

[1] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multi-robot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, Jul. 2006.

[2] G. A. Kaminka, "Autonomous agents research in robotics: A report from the trenches," in *Proc. AAAI Spring Symp. Ser. Designing Intell. Robots*, Stanford, CA, USA, 2012, pp. 27–33.

[3] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004.

[4] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *Int. J. Adv. Robotic Syst.*, vol. 10, no. 12, p. 399, Dec. 2013.

[5] *This Robot Superhero Could Be the Future of Disaster Relief Operations*. Accessed: Apr. 20, 2019. [Online]. Available: https://www.weforum.org/agenda/2018/03/a-six-foot-tall-humanoid-robot-could-come-to-your-rescue-during-a-disaster

[6] *NASA Counting on Humanoid Robots for Deep Space Exploration*. Accessed: Apr. 20, 2019. [Online]. Available: http://www.spaceflightinsider.com/organizations/nasa/nasa-counting-on-humanoid-robots-in-deep-space-exploration/

[7] *Another Robot Just Broke Down Investigating Fukushima's Record High Radiation Levels*. Accessed: Apr. 20, 2019. [Online]. Available: https://www.theverge.com/2017/2/17/14652274/fukushima-nuclear-robot-power-plant-radiation-decomission-tepco

[8] *Ocean Tech: Using Robots to Conduct Deep-Sea Research*. Accessed: Apr. 20, 2019. [Online]. Available: https://oceanbites.org/ocean-tech-using-robots-to-conduct-deep-sea-research/

[9] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[10] M. J. Krieger, J.-B. Billeter, and L. Keller, "Ant-like task allocation and recruitment in cooperative robots," *Nature*, vol. 406, no. 6799, pp. 992–995, Aug. 2000.

[11] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, "Working together: A review on safe human-robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26754–26773, 2017.

[12] T. Zheng and J. Q. Li, "Multi-robot task allocation and scheduling based on fish swarm algorithm," in *Proc. 8th World Congr. Intell. Control Automat.*, Jul. 2010, pp. 6681–6685.

[13] X. Li, Z. Liu, and F. Tan, "Multi-robot task allocation based on cloud ant colony algorithm," in *Proc. Int. Conf. Neural Inf. Process.*, 2017, pp. 3–10.

[14] 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.

[15] A. Rauniyar and P. K. Muhuri, "Multi-robot coalition formation problem: Task allocation with adaptive immigrants based genetic algorithms," in *Proc. IEEE Int. Conf. Syst. Man and Cybern. (SMC)*, Oct. 2016, pp. 137–142.

[16] J. Guerrero, G. Oliver, and O. Valero, "Multi-robot coalitions formation with deadlines: Complexity analysis and solutions," *PLoS ONE*, vol. 12, no. 1, pp. 1–26, Jan. 2017.

[17] A. Hussein and A. Khamis, "Market-based approach to multi-robot task allocation," in *Proc. Int. Conf. Individual Collective Behaviors Robot. (ICBR)*, Dec. 2013, pp. 69–74.

[18] M. J. Matarić, G. S. Sukhatme, and E. H. østergaard, "Multi-robot task allocation in uncertain environments," *Auton. Robots*, vol. 14, nos. 2–3, pp. 255–263, Mar. 2003.

[19] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 25, no. 3, pp. 225–241, Mar. 2006.

[20] M. Strens and N. Windelinckx, "Combining planning with reinforcement learning for multi-robot task allocation," in *Adaptive Agents and Multi-Agent Systems II* (Lecture Notes in Computer Science), D. Kudenko, D. Kazakov and E. Alonso, Eds., Berlin, Germany: Springer-Verlag, 2005, pp. 260–274.

[21] F. Duvallet and A. Stentz, "Imitation learning for task allocation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 3568–3573.

[22] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2110–2216.

[23] M. McIntire, E. Nunes, and M. Gini, "Iterated multi-robot auctions for precedence-constrained task scheduling," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2016, pp. 1078–1086.

[24] E. Nunes, M. Mcintire, and M. Gini, "Decentralized allocation of tasks with temporal and precedence constraints to a team of robots," in *Proc. IEEE Int. Conf. Simulation Modeling Program. Auton. Robots (SIMPAR)*, Dec. 2016, pp. 197–202.

[25] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 932–947, Aug. 2017.

[26] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multi-robot task assignment with task deadline constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 876–888, Jul. 2015.

[27] T. L. Basegio and R. H. Bordini, "Allocating structured tasks in heterogeneous agent teams," *Comput. Intell.*, vol. 35, no. 1, pp. 124–155, Feb. 2019.

[28] L. Vig and J. A. Adams, "Multi-robot coalition formation," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 637–649, Aug. 2006.

[29] Y. Chen, X. J. Mao, F. Hou, Q. Wang, and S. Yang, "Combining re-allocating and re-scheduling for dynamic multi-robot task allocation," in *Proc. IEEE Int. Conf. Syst. Man and Cybern. (SMC)*, Oct. 2016, pp. 395–400.

[30] W. Xiang and H. P. Lee, "Ant colony intelligence in multi-agent dynamic manufacturing scheduling," *Eng. Appl. Artif. Intell.*, vol. 21, no. 1, pp. 73–85, 2008.

[31] S. Giordani, M. Lujak, and F. Martinelli, "A distributed multi-agent production planning and scheduling framework for mobile robots," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 19–30, Jan. 2013.

[32] E. G. Jones, M. B. Dias, and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," *Auton. Robots*, vol. 30, no. 1, pp. 41–56, Jan. 2011.

[33] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, "A distributed task allocation algorithm for a multi-robot system in healthcare facilities," *J. Intell. Robotic Syst.*, vol. 80, no. 1, pp. 33–58, Oct. 2015.

[34] D. Claes, F. Oliehoek, H. Baier, and K. Tuyls, "Decentralised online planning for multi-robot warehouse commissioning," in *Proc. 16th Conf. Auton. Agents MultiAgent Syst.*, 2017, pp. 492–500.

[35] C. Liu and A. Kroll, "Memetic algorithms for optimal task allocation in multi-robot systems for inspection problems with cooperative tasks," *Soft Comput.*, vol. 19, no. 3, pp. 567–584, Mar. 2015.

[36] C. Liu and A. Kroll, "Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems," *Springerplus*, vol. 5, p. 1361, Aug. 2016.

[37] K. Jose and D. K. Pratihar, "Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods," *Robot. Auton. Syst.*, vol. 80, pp. 34–42, Jun. 2016.

[38] N. Li, C. Remeikas, Y. Xu, S. Jayasuriya, and R. Ehsani, "Task assignment and trajectory planning algorithm for a class of cooperative agricultural robots," *J. Dyn. Syst. Meas. Control*, vol. 137, no. 5, May 2015, Art. no. 051004.

[39] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artif. Intell.*, vol. 99, no. 1, pp. 21–71, Feb. 1998.

[40] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robot. Auton. Syst.*, vol. 23, no. 3, pp. 125–152, Apr. 1998.

[41] S. M. LaValle, "Robot motion planning: A game-theoretic foundation," *Algorithmica*, vol. 26, nos. 3–4, pp. 430–465, Apr. 2000.

[42] T. Simeon, S. Leroy, and J.-P. Lauumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Trans. Robot. Autom.*, vol. 18, no. 1, pp. 42–49, Feb. 2002.

[43] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A review of global path planning methods for occupancy grid maps regardless of obstacle density," *J. Intell. Robot. Syst.*, vol. 84, nos. 1–4, pp. 829–858, Dec. 2016.

[44] D. Spensieri, J. S. Carlson, F. Ekstedt, and R. Bohlin, "An iterative approach for collision free routing and scheduling in multirobot stations," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 950–962, Apr. 2016.

[45] C. Wei, K. V. Hindriks, and C. M. Jonker, "Altruistic coordination for multi-robot cooperative pathfinding," *Appl. Intell.*, vol. 44, no. 2, pp. 269–281, Mar. 2016.

[46] 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.

[47] I. Draganjac, D. Miklić, Z. Kovačić, G. Vasiljević, and S. Bogdan, "Decentralized control of multi-AGV systems in autonomous warehousing applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1433–1447, Oct. 2016.

[48] H. G. Zhang and J. Zhou, "Dynamic multiscale region search algorithm using vitality selection for traveling salesman problem," *Expert Syst. With Appl.*, vol. 60, pp. 81–95, Oct. 2016.

[49] S. Öztürk and A. E. Kuzucuoğlu, "Optimal bid valuation using path finding for multi-robot task allocation," *J. Intell. Manuf.*, vol. 26, no. 5, pp. 1049–1062, Oct. 2015.

[50] A. R. Mosteo and L. Montano, "Comparative experiments on optimization criteria and algorithms for auction based multi-robot task allocation," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 2007, pp. 3345–3350.

[51] A. Farinelli, L. Locchi, and D. Nardi, "Distributed on-line dynamic task assignment for multi-robot patrolling," *Auto. Robots*, vol. 41, no. 6, pp. 1321–1345, Aug. 2017.

[52] N. Ayanian, D. Rus, and V. Kumar, "Decentralized multirobot control in partially known environments with dynamic task reassignment," *IFAC Proc. Volumes*, vol. 45, no. 26, pp. 311–316, Sep. 2012.

[53] L.-N. Zu, Y.-T. Tian, J.-C. Fu, and J.-F. Liu, "Algorithm of task-allocation based on realizing at the lowest cost in multimobile robot system," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Aug. 2004, pp. 152–156.

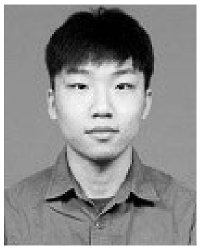
[54] H. Zhou, M. Song, and W. Pedrycz, "A comparative study of improved GA and PSO in solving multiple traveling salesmen problem," *Appl. Soft Comput.*, vol. 64, pp. 564–580, Mar. 2018.

[55] B. Soylu, "A general variable neighborhood search heuristic for multiple traveling salesmen problem," *Comput. Ind. Eng.*, vol. 90, pp. 390–401, Dec. 2015.

[56] S.-M. Chen and C.-Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Syst. Appl.*, vol. 38, no. 12, pp. 14439–14450, 2011.

HONGGUANG ZHANG received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology, in 2000, 2002, and 2005, respectively. From 2006 to 2010, he was a Test Engineer with the China Academy of Space Technology. He was an Associate Professor with the Beijing University of Posts and Telecommunications. He is currently with the Beijing University of Posts and Telecommunications. His research interests include TSP, fire rescue optimization, real optimization algorithms, online optimization systems, and intelligent communications. He was a recipient of the First Prize Award of Science and Technology of Electronic Information from the Chinese Institute of Electronics.

HAN LUO is currently pursuing the M.S. degree with the Beijing University of Posts and Telecommunications. His research interest includes task allocation.



ZAN WANG is currently pursuing the M.S. degree with the Beijing University of Posts and Telecommunications. His research interest includes the flow control of intelligent bus.

YUHONG LIU is currently pursuing the M.S. degree with the Beijing University of Posts and Telecommunications. His research interest includes intelligent protocols.

YUANAN LIU was a Professor with the Beijing University of Posts and Telecommunications. His research interests include 5G systems and electromagnetic interference smart antennas.