Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 32(1): 445–472.
DOI: 10.3934/era.2024022
Received: 29 September 2023
Revised: 01 December 2023
Accepted: 19 December 2023
Published: 28 December 2023

Research article

A multi-strategy genetic algorithm for solving multi-point dynamic aggregation problems with priority relationships of tasks

Yu Shen[1] **and Hecheng Li**[2,*]

[1] School of Computer Science and Technology, Qinghai Normal University, Xining 810008, China
[2] School of Mathematics and Statistics, Qinghai Normal University, Xining 810008, China

* **Correspondence:** Email: lihecheng@qhnu.edu.cn.

Abstract: The multi-point dynamic aggregation problem (MPDAP) that arises in practical applications is characterized by a group of robots that have to cooperate in executing a set of tasks distributed over multiple locations, in which the demand for each task grows over time. To minimize the completion time of all tasks, one needs to schedule the robots and plan the routes. Hence, the problem is essentially a combinatorial optimization problem. The manuscript presented a new MPDAP in which the priority of the task was considered that is to say, some tasks must be first completed before others begin to be executed. When the tasks were located at different priority levels, some additional constraints were added to express the priorities of tasks. Since route selection of robots depends on the priorities of tasks, these additional constraints caused the presented MPDAP to be more complex than ever. To efficiently solve this problem, an improved optimization algorithm, called the multi-strategy genetic algorithm (MSGA), was developed. First of all, a two-stage hybrid matrix coding scheme was proposed based on the priorities of tasks, then to generate more route combinations, a hybrid crossover operator based on 0-1 matrix operations was proposed. Furthermore, to improve the feasibility of individuals, a repair schedule was designed based on constraints. Meanwhile, a q-tournament selection operator was adopted so that better individuals can be kept into the next generation. Finally, experimental results showed that the proposed algorithm is feasible and effective for solving the MPDAP.

Keywords: multi-point dynamic aggregation problem; multi-robot system; task allocation; genetic algorithm; priority

1. Introduction

The multi-point dynamic aggregation problem (MPDAP) [1] is a topic of great interest in the field of optimization. It is widely applied in real-world applications, such as intelligent security [2], agricultural production [3], post-disaster search [4], resource allocation and scheduling for medical care [5, 6], multi-robot target detection and tracking [7], e-commerce logistics system [8] and other task allocation optimization applications [9]. In a nutshell, the MPDAP is that multiple tasks are distributed in different locations, the demand of each task is increasing over time and a group of robots goes to execute these tasks from the initial position so that the time to complete the tasks is minimized.

MPDAP originated as a multi-robot task scheduling system, which determines how robots are scheduled to cooperatively complete tasks in order to optimize the performance of a multi-robot task scheduling system [10, 11]. Due to the complex relationship between robots and tasks caused by time-varying task demands and the coordinated execution of tasks, MPDAP is a very challenging and interesting problem. Over the past few years, researchers have proposed several approaches to solve this problem. In [12], Gao et al. proposed an adaptive cooperative ant colony optimization algorithm (AC-ACO). In the framework of the ant colony algorithm, a new pheromone matrix and a pheromone updating mechanism were proposed to improve the efficiency of solution construction; meanwhile, in order to reduce the search space and remove some bad solutions, a pheromone restoration based mechanism was investigated to improve the ability of constructing feasible solutions, and local search was used to improve the balance between exploration and exploitation [12]. With the purpose of making each robot effectively avoid collisions with obstacles and other robots in the environment, Xin et al. [1] proposed a distributed motion planning algorithm for motion path planning of multiple robots. Simulation results show that the algorithm can effectively solve the motion planning problem of multiple robots in the MPDAP task and can cooperate to efficiently accomplish the MPDAP task. Xin et al. [13] abstracted and modeled forest fire suppression as an MPDAP. For this problem, an estimation of distribution algorithm (EDA) using K-means clustering and multi-modal Gaussian distribution was designed, in which a multi-mutation encoding/decoding method was employed. The experimental results show that the proposed EDA can find high quality solutions, and outperform genetic algorithms and random search methods. For the agent routing problem in multi-point dynamic tasks (ARP-MPDT), Lu et al. [14] used a multi-model estimation distribution algorithm with node histogram model (NHM) and edge histogram model (EHM) in probabilistic model, in which the selection ratio of NHM and EHM probability models was adaptively adjusted. The effectiveness and stability of the algorithm was demonstrated through comparative experiments. Gao et al. [15] designed two memetic algorithms based on different individual learning strategies to improve the search capability of the algorithm for solving MPDAP, which were equality one-step local search (MA-OLS) with better exploration capability and elite multi-step local search (MA-MLS) with better exploitation capability. Experimental results show that the proposed memetic algorithms outperform the state of the art method in solving the task planning problems of MPDA. In order to efficiently solve the MPDAP, a hybrid differential evolution (DE) and distribution estimation algorithm (EDA) called DE-EDA was proposed in [16]. The algorithm combined the advantages of the differential evolution algorithm and distribution estimation algorithm. DE-EDA was also applied to several MPDAP instances of different sizes and compared with other methods in terms of convergent speed and solution quality, respectively. The results show that DE-EDA can effectively solve the MPDAP. In [17], Chen et al. studied multi-agent dynamic task allocation based on a forest fire point model and established a fire spread and dynamic task allocation model. Chen et al. proposed a dynamic task allocation scheme based on global information, which ensured that each reallocation can shorten the task completion time and make all task completion times close to each other. Experiments can verify the better performance of the proposed dynamic task allocation scheme. In [18], Yuan et al. proposed a multi-agent deep reinforcement learning-based multi-robot task assignment method, which has not only the advantage of reinforcement learning to deal with dynamic environments, but also utilizes deep learning to solve the task assignment problem with large state space and high complexity.

These algorithms mentioned above can efficiently solve traditional MPDAP. However, few researches involve the priority of the tasks, and the existing approaches cannot be directly used to deal with this kind of MPDAPs with priority constraints. In this manuscript, we focus on solving MPDAP with task priority constraints (MPDATP). In a real practical environment, some tasks need to be prioritized according to urgent degree due to special locations of these tasks, which may lead to greater losses. According to this fact, when the tasks are executed, one not only needs to consider the total completion time but also, and more importantly, the urgency degree of the tasks. So, the urgent tasks with priority should be completed before others. Such as in a fire scene, there are some fire points surrounded by chemical plants or densely populated places, then these fire points must be priorly disposed. For example, Figure 1 shows a multi-robot fire-fighting mission with priority relationships of tasks. Three robots are assigned to execute 5 tasks. Tasks 2 and 3 are located near the school and chemical plant, respectively; hence, these two tasks need to be dealt with first. In MPDATP, the priorities of partial tasks are considered according to the environment. The priority relationships of the tasks in Figure 1 can be expressed as $task_2 \rightarrow task_3 \rightarrow [task_1, task_4, task_5]$; that is to say, $ct_2 < ct_3 < ct_i$ $(i = 1, 4, 5)$, where ct_i are the completion times of task i $(i = 1, 2, 3, 4, 5)$.

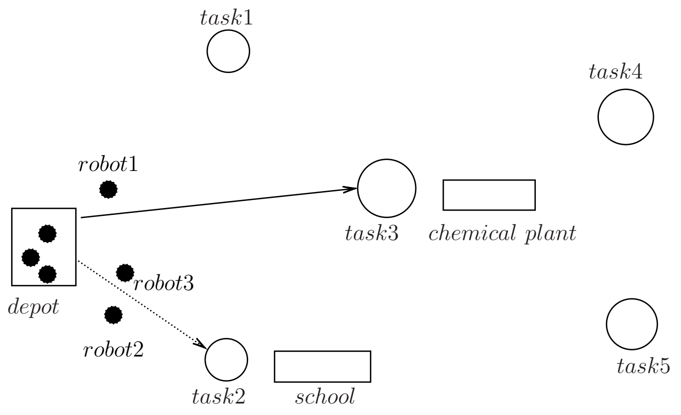


Figure 1. The multi-robot fire-fighting mission with priority relationships of tasks.

In this case, in order to avoid making loss greater, the priority relationships of tasks need to be considered in this set of tasks. Such fire points need to be prioritized for fire fighting and rescue, so in this manuscript, the priority relationships of task execution are taken into account. In a practice setting, it is reasonable to take the order of task completion as additional constraints when the urgency of the task is taken into account. However, it can increase the difficulty of searching efficient scheduling solutions. This is because when scheduling the robots to complete this set of tasks, it is not only required to satisfy the priority order constraints, but also to minimize the completion time of the tasks.

Thus the MPDATP is a challenging problem. Based on the varying urgency of the tasks, we consider the priority relationships of tasks and the original set of tasks can be divided into two groups of tasks: One with priority relationships and another without. Therefore, the robots need to execute two groups of tasks. The decision maker needs to make decisions on these two groups of tasks according to the priority. Additionally, the robots are cooperative in completing this set of tasks, which causes a large search load compared to the general noncooperative one. Finally, the demand of each task is increasing over time, and a small delay on a task can produce large task completion times. In addition, high-quality evolutionary operators are also required to produce more good scheduling solutions for the complex combinatorial optimization problem.

It should be noted that most of the existing models for MPDAPs are to minimize the completion time of all tasks, in which each task has the same priority level. However, in a practice setting, a set of tasks has different priority levels. A recurring question is "When tasks have different levels of urgency, how should the decision maker go about scheduling the robots to efficiently accomplish this set of tasks?"; that is to say, how can the decision maker schedule the robots to complete the set of tasks in a way that satisfies the priority constraints while minimizing the time to complete the tasks, given the priority of the tasks? Different from the above literatures, in this manuscript a new optimization problem, MPDATP, is provided, which is solved by designing a new encoding scheme and efficient evolutionary operators. The main innovations of this manuscript are presented as follows:

1) Based on the real-world requirement, a new MPDATP model is provided by adding priority constraints to the original one. In the proposed model, some tasks need to be completed before others due to priority, which has a different optimization procedure from traditional models.

2) A new two-stage hybrid matrix coding approach is proposed based on priority relationships of the tasks. The first stage is designed to optimize the routes of robots with priority tasks, whereas the second stage is for other tasks.

3) To generate more combinations of routes for the robots to execute the tasks, the crossover operator using matrix operations is adopted. Moreover, to improve the feasibility of individuals, two efficient repair operators are designed for infeasible solutions.

The rest of the manuscript is arranged as follows. Sections 2 and 3 introduce the related research and the MPDATP model, respectively. The detailed algorithm design is presented in Section 4. The experimental studies are implemented in Section 5. Finally, Section 6 concludes this manuscript.

2. Related research

Evolutionary algorithms simulate the evolution of organisms in nature, because nature has a magical power to preserve good genes and evolve strong genes that are more suitable for survival. By comparing with traditional optimization algorithms, evolutionary computation is a mature global optimization method with high robustness and wide applicability, which has the characteristics of self-organization, self-adaptation and self-learning. Furthermore, it is able to deal with complex problems that are difficult to be solved by traditional optimization algorithms.

Evolutionary algorithms have a particularly wide range of applications, so many researchers use different evolutionary algorithms to solve various of complex optimization problems in practice [19], such as unmanned aerial vehicle (UAV) path planning [20], flexible job shop scheduling [21] and flight operation data-sharing [22]. There are also a number of other applications available in the literature [23–25].

The genetic algorithm is one of the most popular evolutionary algorithms and it is widely used in many fields. For the cooperation and collision avoidance problem of multiple unmanned surface vehicles, Wang et al. [26] proposed an improved genetic algorithm as the core algorithm for planning collision avoidance, which was improved by retaining, deleting and replacing, using hierarchical analysis to establish the degree of fit and iteratively optimizing the adjustment of the speed and heading to calculate the optimal collision avoidance paths for the current multiple unmanned surface vehicles. Milad et al. [27] proposed an enhanced genetic algorithm to plan the paths of multiple mobile robots in a continuous environment, and five customized crossover and mutation operators were used to improve the initial paths and to find the optimal path between the start position and the destination. For the problem of autonomous navigation and control of unmanned ground vehicles, Xin et al. [28] proposed a genetic algorithm based on a multi-domain inversion strategy, which can help to effectively improve the ability of localized search and increase the probability of producing good individuals. In [29], a quantitative particle swarm algorithm and a multi-population genetic algorithm were combined to construct a new hybrid algorithm, which can effectively solve the automated dock two-level scheduling model. For multi-person surface vehicle systems, Xia et al. [30] proposed an improved self-organizing mapping and an improved genetic algorithm to solve multitask assignment and path planning. In the problem of collision-free shortest path planning for mobile agents, Lee et al. [31] proposed a method that combined a genetic algorithm and a directional factor to the target point. Additionally, other interesting applications can be found in [32–35]. Due to the high performance of genetic algorithms for solving complex combinatorial optimization problems, we adopt the genetic algorithm to solve MPDATP in this manuscript.

3. The MPDATP model

In the MPDATP, a set of tasks $S_{tasks} = \{task_0, task_1, task_2, ..., task_N\}$ and a set of robots $S_{robots} = \{robot_1, robot_2, ..., robot_M\}$ are considered. Each $robot_k$ executes tasks with a work efficiency v^k and their initial locations are in the depot (denoted as $task_0$). Each $task_i$ has an inherent time-varying demand $q_i(t)$, and the relationship between task demand and time is suggested by the following equations in [12]:

$$q_i(t) = q_i(0) + \beta_i t\,, \tag{3.1}$$

where $q_i(t)$ represents the demand of $task_i$ at time t, the initial demand of $task_i$ is $q_i(0)$ and β_i is the inherent increment rate of $task_i$.

The MPDATP model is described as follows:

$$min\, f = \max_{i=1,2,\cdots,N} ct_i \tag{3.2}$$

subject to

$$\sum_{j=0}^{N} y_{ij}^k = \sum_{j=0}^{N} y_{ji}^k \qquad \forall i = 1,2,\cdots,N \qquad \forall k = 1,2,\cdots,M \tag{3.3}$$

$$\sum_{i=0}^{N}\sum_{k=1}^{M} y_{ij}^k \geq 1 \qquad \forall j = 1,2,\cdots,N \tag{3.4}$$

$$\sum_{i=0}^{N} y_{ij}^k \leq 1 \qquad \forall j = 1,2,\cdots,N \qquad \forall k = 1,2,\cdots,M \tag{3.5}$$

$$at_0^k = ct_0 = 0 \qquad \forall k = 1,2,\cdots,M \tag{3.6}$$

$$at_j^k = \sum_{i=0}^{N}(ct_i + t_{ij})y_{ij}^k \qquad \forall j = 1, 2, \cdots, N \qquad \forall k = 1, 2, \cdots, M \tag{3.7}$$

$$q_j(t) - [\sum_{i=0}^{N}\sum_{k=1}^{M}(ct_j - at_j^k)y_{ij}^k v^k] = 0 \qquad \forall j = 1, 2, \cdots, N \tag{3.8}$$

$$\beta_j < \sum_{k=1}^{M}\sum_{i=0}^{N} y_{ij}^k v^k \qquad \forall j = 1, 2, \cdots, N \tag{3.9}$$

$$y_{ij}^k \in \{0, 1\}, i \neq j, \qquad \forall j = 1, 2, \cdots, N \qquad \forall k = 1, 2, \cdots, M \tag{3.10}$$

$$ct_i < ct_j, if\ the\ priority\ of\ task\ i\ is\ higher\ than\ that\ of\ task\ j\ , \tag{3.11}$$

where the binary decision variables y_{ij}^k is taken as 1 if robot k goes from task i to task j, and 0 otherwise. The notations used in the formulations are summarized in Table 1.

Table 1. Notations used in the problem formulation.

Notation	Description
ct_i	the completion time of task i
N	the number of tasks
M	the number of robots
at_j^k	the arrival time of robot k at task j
t_{ij}	the travel time from task i to task j
$q_j(t)$	the demand of task j accumulate at time t
v^k	the ability of of robot k
β_j	the inherent increment rate of task j

The objective (3.2) is to minimize the maximum completion time of all the tasks. However, it is worth noting that in MPDATP, the objective function does not include the time spent returning to the depot. Constraint (3.3) means that each robot has the same number of paths into each task as the number of paths out of the task. Constraint (3.4) ensures that each task is executed by at least one robot. Constraint (3.5) indicates that each task is executed by each robot at most once. Constraint (3.6) sets the arrival time and completion time for the depot to 0. Constraint (3.7) implies that a robot immediately goes to its next task once it completes its current task. Constraint (3.8) indicates that a task is completed when its demand decreases to zero (i.e., the accumulated demand from time 0 to ct_j equals to the total demand reduced by the robots executing the task during this time period). Constraint (3.9) implies that for each task, the total ability of the robots executing the task must be greater than the inherent increment rate of the task so that the task can be completed. Constraint (3.10) sets the binary domain of the decision variable. Constraint (3.11) shows the effect of task priority on task completion time.

4. Algorithmic design

In this section, the multi-strategy genetic algorithm (MSGA) proposed in this manuscript is presented. Since a set of tasks is divided into two groups based on whether they have priority relationships or not, the encoding is coded in a two-stage mixed 0–1 matrix with different decoding methods for different groups. The three crossover operators are designed based on addition, subtraction and the dot product of matrices to generate more route combinations and potentially good solutions. The individual repair operators are designed based on the inverse order number in mathematics to rationalize the inverse order of the task completion time so as to satisfy the priority relationships of the task completion time, as well as to improve the feasibility of the solution. Finally, the pseudo-code of MSGA is given and algorithm performance is analyzed.

4.1. Encoding/Decoding strategies

In an MPDATP tasks planning problem with N tasks and M robots, without any loss of generality, the stage I tasks are taken as those with priority. A feasible solution is encoded as an $M \times (N + 1)$ matrix X:

$$X_m = \left[\begin{array}{cccc|cc} 1 & x_{11}^m & x_{12}^m & \cdots & \cdots & x_{1N}^m \\ 1 & x_{21}^m & x_{22}^m & \cdots & \cdots & x_{2N}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cdots & \cdots & x_{rI}^m & \cdots & x_{rN}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{M1}^m & x_{M2}^m & \cdots & \cdots & x_{MN}^m \end{array}\right], \tag{4.1}$$

where the elements in the first column of the matrix (4.1) are all 1, indicating that every robot starts from the depot. From the second column to the last column, the rest of the elements are 0 or 1, as shown in (4.2). (4.3) indicates that for each task, the total ability of the robots executing it must be greater than its inherent increment rate. Otherwise, the task can never be completed. In the matrix (4.1), the left and right sides of the vertical lines indicate the stages I and II, respectively. The tasks with priorities are encoded in the stage I, and the other tasks are encoded in the stage II. It means that there are I tasks with priorities among N tasks, and the remaining $N - I$ tasks do not have. We assume that, according to the environment and the urgency of the tasks, etc., the tasks are ordered from the highest to the lowest priorities. The task with the highest priority is assigned the integer I, and the tasks are assigned the integer in turn until the task with the lowest priority is assigned the integer 1.

$$x_{ri} = \begin{cases} 1, if\ robot\ r\ goes\ to\ task\ i\ and\ completes\ the\ task\ i \\ 0, otherwise \end{cases} \tag{4.2}$$

$$\sum_{r=1}^{M} x_{ri} v^r > \beta_i \,. \tag{4.3}$$

The decoding procedure is executed as follows:

In each row of (4.1), 1 means the robot visits the task, whereas 0 represents the robot never visits the task. In the stage I, the order of visiting tasks is probabilistic according to the priority of the task, and the higher the $p_{task_{k_i}}$ ($k_i \in \{1, 2, \cdots, I\}$), the more likely it is to be visited first.

$$p_{task_{k_i}} = \frac{i}{\sum_{i=1}^{I} i} \tag{4.4}$$

In the stage II, the order of visiting tasks is determined by

$$VO_{task_i} = \frac{\beta_i}{s_{path_i}}, \tag{4.5}$$

where β_i is the inherent increment rate of task i and s_{path_i} is the path length from depot to task i. A task with a larger β_i and a smaller s_{path_i} should be visited preferentially. Hence, VO_{task_i} can be taken as an index by which the tasks can be chosen by a probability choice scheme, one by one.

For example, the example in Figure 1 can be encoded as follows:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{array}\right], \tag{4.6}$$

where

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.7}$$

and

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{4.8}$$

are the encodings for stages I and II, respectively. For narrative clarity, matrix (4.6) is written as matrix (4.9).

$$\left[\begin{array}{cccc|ccc} robot & depot & task_2 & task_3 & task_1 & task_4 & task_5 \\ robot_1 & 1 & 0 & 1 & 1 & 1 & 0 \\ robot_2 & 1 & 1 & 1 & 0 & 1 & 1 \\ robot_3 & 1 & 1 & 1 & 0 & 1 & 1 \end{array}\right] \tag{4.9}$$

In matrix (4.9), the first row [1 0 1 1 1 0] means that tasks [$depot, task_3, task_1, task_4$] need to be visited sequentially. The values of the relevant parameters of the tasks are shown in Tables 2 and 3. Based on the visiting probability value $p_{task_3} > p_{task_1}, p_{task_3} > p_{task_4}$ and the order $VO_{task_4} > VO_{task_1}$, the visiting path of $robot_1$ is determined as $depot \to task_3 \to task_4 \to task_1$.

Table 2. The values of p_{task} of the tasks in Figure 1.

Parameters	$task_2$	$task_3$	$task_1$	$task_4$	$task_5$
i	2	1	0	0	0
p_{task}	2/3	1/3	0	0	0

Table 3. The values of VO_{task} of the tasks in Figure 1.

Parameters	$task_1$	$task_4$	$task_5$
β	0.3	1.5	1.2
s	3	5	6
VO_{task}	0.1	0.3	0.2

4.2. Evolutionary operators

I. Crossover operator

The crossover operator is a genetic recombination process that mimics reproduction in nature, in which the individual quality of the population is improved. In order to improve the performance of individuals and produce more promising individuals, three crossover operators are proposed.

Individuals participating in the crossover are selected with crossover probability p_c and the selected individuals are randomly paired. For each pair of parents, three crossover operators are randomly selected to produce two offspring and these two offspring are used to replace the parental points, and the points that do not participate in the crossover are directly regarded as their own offspring, thus obtaining the set of crossover offspring.

Crossover operator 1, matrix addition strategy:

$$O_{c_1} = X_m + X_n \, . \tag{4.10}$$

The offspring of the crossover operator 1 can be described as

$$O_{c_1} = \left[\begin{array}{cccc|cc} 1 & x_{11}^m + x_{11}^n & x_{12}^m + x_{12}^n & \cdots & \cdots & x_{1N}^m + x_{1N}^n \\ 1 & x_{21}^m + x_{21}^n & x_{22}^m + x_{22}^n & \cdots & \cdots & x_{2N}^m + x_{2N}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cdots & \cdots & x_{rI}^m + x_{rI}^n & \cdots & x_{rN}^m + x_{rN}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{M1}^m + x_{M1}^n & x_{M2}^m + x_{M2}^n & \cdots & \cdots & x_{MN}^m + x_{MN}^n \end{array}\right] . \tag{4.11}$$

Crossover operator 2, matrix subtraction strategy:

$$O_{c_2} = X_m - X_n \, . \tag{4.12}$$

The offspring of the crossover operator 2 can be described as

$$O_{c_2} = \left[\begin{array}{cccc|cc} 1 & x_{11}^m - x_{11}^n & x_{12}^m - x_{12}^n & \cdots & \cdots & x_{1N}^m - x_{1N}^n \\ 1 & x_{21}^m - x_{21}^n & x_{22}^m - x_{22}^n & \cdots & \cdots & x_{2N}^m - x_{2N}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cdots & \cdots & x_{rI}^m - x_{rI}^n & \cdots & x_{rN}^m - x_{rN}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{M1}^m - x_{M1}^n & x_{M2}^m - x_{M2}^n & \cdots & \cdots & x_{MN}^m - x_{MN}^n \end{array}\right] . \tag{4.13}$$

Crossover operator 3, corresponding element multiplication strategy:

$$O_{c_3} = X_m * X_n \, . \tag{4.14}$$

The offspring of the crossover operator 3 can be described as

$$O_{c_3} = \left[\begin{array}{cccc|cc} 1 & x_{11}^m \times x_{11}^n & x_{12}^m \times x_{12}^n & \cdots & \cdots & x_{1N}^m \times x_{1N}^n \\ 1 & x_{21}^m \times x_{21}^n & x_{22}^m \times x_{22}^n & \cdots & \cdots & x_{2N}^m \times x_{2N}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cdots & \cdots & x_{rI}^m \times x_{rI}^n & \cdots & x_{rN}^m \times x_{rN}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{M1}^m \times x_{M1}^n & x_{M2}^m \times x_{M2}^n & \cdots & \cdots & x_{MN}^m \times x_{MN}^n \end{array}\right]. \tag{4.15}$$

Of the above three crossover operators, crossover operators 1 and 2 are designed based on the addition and subtraction operations of matrices and crossover operator 3 is a newly defined matrix multiplication rule for multiplying elements in corresponding positions in a matrix. The use of different crossover operators may generate more potential routes, which can improve the diversity of the population.

II. Mutation operator

The mutation operator can improve the diversity of populations to some extent and likewise contributes to the feasibility of populations. In order to produce potentially good solutions, the mutation operation with 0-1 reciprocal change is used.

First, a fixed mutation probability p_m and a random number η are generated for each gene in each offspring in the crossover offspring set. If $\eta \leq p_m$, the number r in the bit is changed to $1 - r$; otherwise, the number remains unchanged. Specifically, the number r is changed into another number $1 - r$; otherwise, the bit remains unchanged. The mutation operator is performed on the elements of all columns except for the first column.

III. Individual repair operator

During the evolution process, it may happen that robots cannot complete the tasks assigned to them or they can complete the tasks but the time to complete them does not satisfy the priority restriction, which leads to an infeasible solution. In MPDATP, it should first be rationalized that the robot can complete each task. In addition, the priority relationship needs to be satisfied for each task. Finally, it is desired that the time to complete the task is minimized. An efficient scheduling scheme requires to adjust the robot's routes such that the priority constraints are satisfied, resulting in a high-quality feasible solution.

In the individual matrix (4.1), the elements in the first column are all 1, but the elements in the other positions may not take values of 0 or 1, and perhaps do not satisfy the constraint (4.3). That is, there are no robots to execute a certain task or the scheduled robots cannot complete the task, so the repair operator simply adds robots to this task until they can complete this task. For constraint (4.3), the repair operator 1 is used, which is given as follows:

Step 1. The elements in the individual matrix (4.1) are made feasible, i.e., the elements greater than 1 become 1, the elements less than 0 become 0.

Step 2. Verify whether the elements of each column satisfy constraints (4.3); if so, turn to Step 4; otherwise, turn to Step 3.

Step 3. Randomly select the element in each column that is 0 and change it to 1 until the element in that column satisfies constraint (4.3).

Step 4. End.

In the MPDATP, tasks with priorities are also considered. The fact that individuals satisfy constraint (4.3) just means that the set of tasks can be accomplished by these robots, but there is a possibility that they do not satisfy the priority relationship of the tasks, so it is necessary to replan the routes of some robots, and the planning makes it possible for the robots not only to accomplish this set of tasks, but also to satisfy the priority relationship of the tasks. Therefore, we need to continue to optimize the routes of robots so that it satisfies constraint (3.11) by individual repair operator 2. For the sake of narrative convenience, we refer to the tasks that have a priority relationship as the tasks in stage I and the others as the tasks in stage II.

An example is presented to illustrate the individual repair operator 2. The problem has a total of 10 tasks, where 5 tasks have priority relationships and other tasks do not, so it is useful to set their priority relationship as $T_1 < T_2 < T_3 < T_4 < T_5 < T_6, T_7, T_8, T_9, T_{10}$.

Case 1 $T_i (i = 1, 2, 3, 4, 5) < T_j (j = 6, 7, 8, 9, 10)$ is met.

Step 1. If $T_1 < T_2 < T_3 < T_4 < T_5$ is met, the repair process stops; otherwise, go to Step 2.

Step 2. Rank the completion times of stage I tasks to obtain the order $T_3 < T_2 < T_4 < T_1 < T_5$; see Table 4.

Table 4. Task completion time order(I).

Task	$task_1$	$task_2$	$task_3$	$task_4$	$task_5$
Target order	1	2	3	4	5
Inverse order number	3	1	0	0	0
Inverse proportion	3/4	1/4	0	0	0

In Table 4, the target order refers to the order relationship that the task priority needs to be satisfied eventually, and the inverse order number represents the sum of the inverse order of a task in the current order. The current order of the 5 tasks is 32415 ($task_3$, $task_2$, $task_4$, $task_1$, $task_5$); 3 and 2 constitute an inverse order, so the inverse order number of $task_2$ is 1; 3, 2 and 4 all constitute the inverse order to 1, so the inverse order number of $task_1$ is 3.

Step 3. Internally schedule robots.

Since the inverse order number of $task_3$, $task_4$ and $task_5$ is 0, the inverse order numbers of $task_2$ and $task_1$ are 1 and 3, respectively. So, $task_2$ and $task_1$ need to be moved forward by 1 and 3 bits, respectively. Schedule robots from $task_3$ and $task_4$ to $task_2$ or $task_1$ using the $task_5$ completion time as a criterion.

The time to complete $task_3$ is set to be 10 percent less than the time to complete $task_5$, and the time to complete $task_4$ is set to be 5 percent less than the time to complete $task_5$. Under this criterion, other robots are dispatched from $task_3$ and $task_4$ into $task_1$ or $task_2$, and for ease of narration, these robots are made into the Active Robot Group. Since $task_1$ has a larger inverse order number, 75 percent of the robots from the Active Robot Group are randomly scheduled to $task_1$, and the remaining 25 percent of the robots are scheduled to $task_2$. The probability of scheduling a robot is determined by the inverse proportion of the task, e.g., if the inverse order number of $task_1$ is 3 and the inverse order number of $task_2$ is 1, the sum of the inverse order numbers is 4, so three-quarters (i.e., 75 percent) of the robots from the Active Robot Group are scheduled to $task_1$. For experimental convenience, this procedure is executed up to 3 times and is stopped once the constraint (3.11) is satisfied; otherwise, Step 4 is performed.

Step 4. Externally schedule robots.

Schedule robots from the stage II tasks to the stage I tasks according to probability. In order to minimize the total time to complete the tasks, it is specified that the smaller the time to complete a task in the stage II tasks, the higher the probability of scheduling robots from that task, and the robots are randomly scheduled within a given task. The process doesn't stop until the constraint (3.11) is satisfied.

Case 2 T_i $(i = 1, 2, 3, 4, 5) < T_j$ $(j = 6, 7, 8, 9, 10)$ is not met.

Step 1. Rank the completion times of stage I tasks to obtain the order $T_3 < T_4 < T_{i_1} < T_5 < T_{i_2} < T_1 < T_{i_3} < T_{i_4} < T_2$, i_1, i_2, i_3 and $i_4 \in \{6, 7, 8, 9, 10\}$; see Table 5. In case 2, only the order in which the stage I tasks are listed in the overall task set is required.

Table 5. Task completion time order(II).

Task	$task_1$	$task_2$	$task_3$	$task_4$	$task_5$
Target order	1	2	3	4	5
Inverse order number	5	7	0	0	0
Inverse proportion	5/12	7/12	0	0	0

Step 2. Schedule robots from the Active Robot Group.

The current order of $task_1$ and $task_2$ are 6 and 9 in the whole task set, respectively. The $task_3$, $task_4$ and $task_5$ are ranked in the top 5, indicating that before $task_2$, there are 4 stage II tasks that take less time to complete. On the basis of being able to complete these 4 tasks of stage II and for ease of narration, the rest of these robots are made into the Active Robot Group, and the robots from the Active Robot Group are randomly scheduled to $task_1$ or $task_2$. Five-twelfths of the robots from the Active Robot Group are randomly scheduled to $task_1$, and the remaining seven-twelfths of the robots are scheduled to $task_2$. The process of randomly scheduling the remaining robots is executed up to three times. Sort the completion times and stop if constraint (3.11) is satisfied, or, if the case 1 is satisfied, continue the repair process according to case 1; otherwise, proceed to Step 3.

Step 3. Schedule remaining robots from the stage II tasks.

With the criterion that the stage II tasks can be completed, the remaining robots are randomly scheduled to tasks with non-zero inverse ordinal numbers in the stage I task. The process of randomly scheduling the remaining robots is executed up to three times. Sort the completion times and stop if constraint (3.11) is satisfied, or, if the case 1 is satisfied, continue the repair process according to case 1; otherwise, proceed to Step 4.

Step 4. Schedule all robots from certain tasks in the stage II task.

Randomly schedule all robots from a task in the stage II task group to tasks with a non-zero inverse order number in the stage I task to stop if it satisfies constraint (3.11), or, if the case 1 is satisfied, continue the repair process according to the case 1; otherwise, continue to randomly dispatch all robots from two (or three, or four, . . .) tasks in the stage II task group, schedule all robots to the tasks with a non-zero inverse order number in the stage I task so that it satisfies constraint (3.11) or, in case 1, stop.

IV. Selection operator

The selection operator is intended to keep good individuals into the next generation or to lead potential good individuals to appear in the next generation of the population. In this manuscript, the q-tournament selection operator is adopted.

The q individuals are randomly selected from the parent and the offspring populations, the one with the highest fitness value is saved to the next generation and the process is repeatedly executed until the number of individuals in the next generation reaches a predetermined population size.

4.3. Algorithm details

Based on the above-mentioned design, the pseudo-code of the MSGA is presented in Algorithm 1. Algorithm 2 is the pseudo-code of the repair process for an individual.

Algorithm 1 Procedure of the MSGA

```
Require: $T_{max}$: maximum running time; benchmark instance; $NP$:population size;
Ensure: output the set $A$ to get the optimal solution $X^*$
 1: generate initial population $P(T) = \{X_1, X_2, \cdots, X_{NP}\}$ ; $T \leftarrow 0$.
 2: using Algorithm 2 to repair the individuals in the population $P(T)$.
 3: store the optimal individual $X^*$ to the set $A$ according to the fitness value of the individuals in $P(T)$.
 4: while $T \leq T_{max}$ do
 5:     the crossover offspring population $O(T)_c$ is obtained after the crossover operator of population $P(T)$.
 6:     the offspring population $O(T)$ is obtained after the mutation operator of crossover offspring population $O(T)_c$.
 7:     using Algorithm 2 to repair the individuals in the population $O(T)$.
 8:     the next generation population $P(T+1)$ is obtained from population $P(T)$ and offspring population $O(T)$ according to the selection operator.
 9:     update set $A$ according to the fitness value of the individuals in $O(T)$.
10:     $T = T + 1$.
11: end while
```

Algorithm 2 Procedure for repairing individuals

```
Require: current population: $P(T)^c = \{X_1(T)^c, X_2(T)^c, \cdots, X_{NP}(T)^c\}$;
Ensure: repair population: $P(T)^r = \{X_1(T)^r, X_2(T)^r, \cdots, X_{NP}(T)^r\}$;
 1: for $i = 1 : NP$ do
 2:     if $X_i(T)^c$ satisfies constraint (4.3) then
 3:         if $X_i(T)^c$ satisfies constraint (3.11) then
 4:             replace all the illegal elements in the individual $X_i(T)^c$ by 0 to get the repaired individual $X_i(T)^r$;
 5:         else
 6:             individual $X_i(T)^c$ undergoes repair operator 2 to obtain individual $X_i(T)^r$;
 7:         end if
 8:     else
 9:         individual $X_i(T)^c$ undergoes repair operator 1 and repair operator 2 to obtain individual $X_i(T)^r$;
10:     end if
11: end for
```

4.4. Algorithm analyze

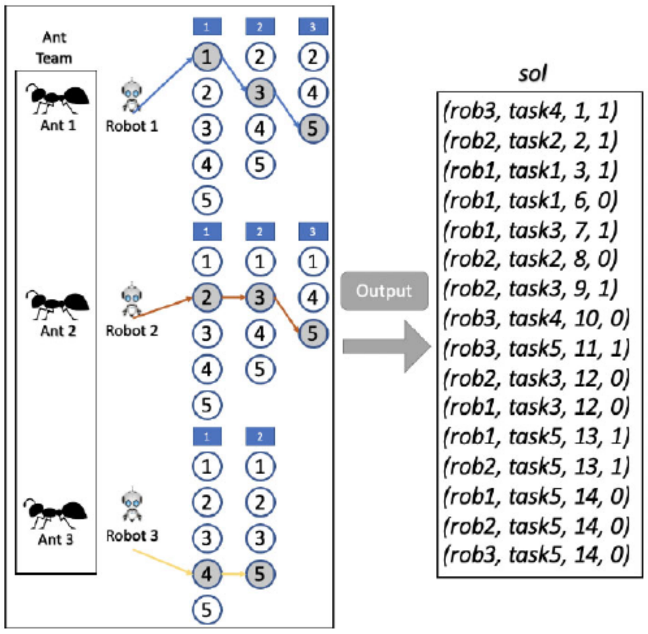
In the original model MPDAP, the priority constraints of the tasks are not considered, so it is only necessary to rationally plan the routes of the robots going to execute the tasks so that the time to complete the set of tasks is minimized. In the MPDATP, some of the tasks have priority constraints, and the obtained rational plan to execute the tasks must satisfy the priority constraints to minimize the time to complete the set of tasks, which makes MPDATP more challenging when compared to the original model. Based on the characteristics of MPDATP, we design MSGA. The reasonableness of the MSGA design is analyzed below.

1) Encoding and decoding strategies

Because some of the tasks in a group of tasks have priority constraints, we use a two-stage hybrid 0-1 matrix encoding. The tasks with priorities are encoded in stage I and the other tasks are encoded in stage II. With the purpose of satisfying the priority constraints, different decoding strategies are used. After decoding, each row is the traveling path of a robot performing a task.

Figure 2 shows the representation of the solution presented in the AC-ACO [12]. According to the instance in Figure 2, as encoded in this manuscript is matrix (4.16).

$$\begin{bmatrix} robot & depot & task_1 & task_2 & task_3 & task_4 & task_5 \\ robot_1 & 1 & 1 & 0 & 1 & 0 & 1 \\ robot_2 & 1 & 0 & 1 & 1 & 0 & 1 \\ robot_3 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.16}$$



(rob3, task4, 1, 1)
(rob2, task2, 2, 1)
(rob1, task1, 3, 1)
(rob1, task1, 6, 0)
(rob1, task3, 7, 1)
(rob2, task2, 8, 0)
(rob2, task3, 9, 1)
(rob3, task4, 10, 0)
(rob3, task5, 11, 1)
(rob2, task3, 12, 0)
(rob1, task3, 12, 0)
(rob1, task5, 13, 1)
(rob2, task5, 13, 1)
(rob1, task5, 14, 0)
(rob2, task5, 14, 0)
(rob3, task5, 14, 0)

Figure 2. Solution construction process in AC-ACO.

Analyzing these two representations of the solutions, the encoding in the manuscript is formally concise and decoded according to a decoding mechanism with information about the inherent increment rate of the task as well as the length of the path, making the path of the robot to perform the task more suitable for the MPDAP.

2) Evolution procedure

The crossover and mutation operators in MSGA are designed to generate more potentially good solutions, and the selection operator is designed to keep the good solutions. Individual repair operators are also designed to enable the generation of more feasible good solutions, and to facilitate the convergence of the algorithm. The evolution procedure is suitable for MPDATP.

3) Algorithm performance

MSGA is designed to solve MPDATP with encoding and decoding strategies, and evolutionary operators based on priority constraints. Meanwhile, the number of priority tasks is not going to affect the performance of the algorithm. It can be analyzed that an increase in the number of tasks with priority constraints leads to an increase in the computational cost of finding a reasonable solution for executing the task when dealing with this part of the task. However the computational cost of dealing with the rest of the task will conversely decrease. The computational cost changes in trade-off with the increase or decrease in the number of priorities, so the new model does not evidently increase the computational cost of the optimization algorithm.

5. Simulation

The results of the simulation experiments are divided into two parts. In the first part, the original model of MPDAP [12] is solved by MSGA and the model doesn't involve priority constraints. According to the experimental results, the Friedman's test and Wilcoxon's test, it can be seen that MSGA is effective in solving this kind of MPDAPs. In the second part, the priority constraints are taken into account and MSGA provides the experimental results for the new model with priority constraints.

5.1. Comparison with other methods

All 50 benchmark instances are taken from [12]. In order to make the experimental results well presented, the benchmark instances are divided into 3 groups, namely, Group 1, Group 2 and Group 3. The differences between groups of benchmark instances are shown in Table 6, where N is the number of tasks and M is the number of robots. All the instances are denoted by their group name, number of robots, number of tasks and the ratio between the sum of all the task inherent rates and the sum of all the robot abilities. For example, benchmark instance 1 is named as $G_1\_5\_4\_0.39$. G_1 denotes the group of the instance, 5 is the number of robots, 4 is the number of tasks and 0.39 is the ratio.

Table 6. Differences between groups of benchmark instances.

Instance	The number of benchmark instances	Range of N	Range of M	Range of $N + M$
Group 1	27	[4,40]	[3,30]	(9,50)
Group 2	6	[10,60]	[15,40]	[50,95)
Group 3	17	[15,120]	[20,120]	[95,180]

In order to make the comparison fair and verify the effectiveness of the MSGA, the parameter settings are taken similar to the benchmark instances, and the stopping criterion is set also the same as competitors. The proposed algorithm is run independently for 30 times, and computational results are recorded in Tables 7–12. The data of MA [15] based on two variants is shown in columns MA-MLS and MA-OLS in Tables 7, 9 and 11. The data of EDA [16] is shown in column EDA in Tables 7, 9 and 11. The data of iterative local search (ILS) [35] is presented in column ILS in Tables 8, 10 and 12. The data of AC-ACO [12] is provided in column AC-ACO in Tables 8, 10 and 12. Column MSGA provides the computational results by MSGA.

In Tables 7–12, the comparison data includes the means and standard deviations of the completion time of tasks, and $*$ means the corresponding method cannot obtain a feasible solution in a limited period of time. The numbers in parentheses indicates the rank-order of all compared approaches on a benchmark instance. The smaller the rank value, the better the algorithm. In addition, some visual comparisons are also provided by Figures 3–8. For example, in Figure 3, in the set of bars of MA-MLS, the performance of MA-MLS compared with other methods, ranked 1 is 4 times, ranked 2 is 13 times, ranked 3 is 9 times, and ranked 4 is 1 time, so the total rank value of all is 61, i.e., $1 \times 4 + 2 \times 13 + 3 \times 9 + 4 \times 1 = 61$.

Table 7. Comparison (I) of experimental results on Group 1.

Instance	MA-MLS		MA-OLS		EDA		MSGA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
G_1_5_4_0.39	1.07E+2(4)	1.9E+0	1.06E+2(3)	1.5E+0	1.05E+2(2)	1.2E-1	1.03E+2(1)	1.5E+0
G_1_5_5_1.66	1.22E+3(3)	4.6E+1	1.17E+3(2)	3.2E+1	1.31E+3(4)	5.9E+1	1.13E+3(1)	2.1E+1
G_1_3_10_1.51	9.52E+2(3)	2.1E+1	9.22E+2(2)	3.6E+1	1.03E+3(4)	2.9E+1	8.76E+2(1)	3.3E+0
G_1_3_15_5.03	1.09E+5(3)	4.4E+4	5.98E+4(2)	1.3E+4	∗(4)	∗	2.76E+4(1)	8.3E+2
G_1_5_10_0.93	4.33E+2(2)	1.1E+1	4.20E+2(1)	1.2E+1	4.78E+2(3)	9.4E+0	4.09E+2(1)	6.5E+0
G_1_10_5_1.39	6.00E+2(3)	2.2E+1	5.87E+2(2)	2.6E+1	6.53E+2(4)	2.7E+1	5.65E+2(1)	2.1E+1
G_1_5_10_3.67	3.12E+4(3)	6.3E+3	2.23E+4(2)	2.9E+3	∗(4)	∗	1.30E+4(1)	3.1E+3
G_1_5_20_4.36	5.24E+4(3)	7.2E+3	3.74E+4(2)	4.3E+3	∗(4)	∗	2.03E+3(1)	4.3E+2
G_1_10_10_3.79	3.53E+4(3)	7.8E+3	3.19E+4(2)	6.8E+3	∗(4)	∗	7.79E+3(1)	3.1E+2
G_1_11_11_1.28	3.05E+2(2)	1.7E+1	3.10E+2(3)	1.0E+1	4.06E+2(4)	1.7E+1	2.50E+2(1)	2.0E+1
G_1_30_5_0.46	1.50E+2(1)	9.7E-1	1.50E+2(1)	6.9E-1	1.55E+2(2)	1.7E+0	1.50E+2(1)	1.1E+0
G_1_15_10_1.17	3.39E+2(2)	1.0E+1	3.59E+2(3)	9.0E+0	4.19E+2(4)	1.1E+1	3.17E+2(1)	1.2E+0
G_1_10_15_1.3	6.28E+2(2)	9.7E+0	6.44E+2(3)	1.0E+1	8.07E+2(4)	3.3E+1	6.07E+2(1)	9.7E+0
G_1_20_10_0.47	1.04E+2(3)	1.9E+0	1.03E+2(2)	1.7E+0	1.13E+2(4)	2.5E+0	9.77E+1(1)	1.8E+0
G_1_20_10_0.96	3.41E+2(2)	7.0E+0	3.61E+2(3)	4.5E+0	4.05E+2(4)	1.2E+1	3.31E+2(1)	4.0E+0
G_1_20_10_0.94	2.73E+2(2)	8.5E+0	2.79E+2(3)	3.7E+0	3.12E+2(4)	6.5E+0	2.39E+2(1)	4.5E+0
G_1_5_40_3.95	1.51E+4(3)	7.3E+2	1.43E+4(2)	5.1E+2	2.73E+4(4)	2.2E+3	1.23E+4(1)	4.6E+2
G_1_10_20_6.04	∗(2)	∗	∗(2)	∗	∗(2)	∗	9.10E+4(1)	5.12E+3
G_1_30_10_0.65	1.83E+2(2)	3.9E+0	1.84E+2(3)	3.4E+0	1.97E+2(4)	4.6E+0	1.68E+2(1)	2.4E+0
G_1_15_20_0.67	3.51E+2(1)	7.0E+0	3.67E+2(3)	1.8E+0	3.92E+2(4)	6.1E+0	3.60E+2(2)	3.1E+0
G_1_30_10_1.34	4.14E+2(2)	1.6E+1	4.58E+2(3)	8.0E+0	5.35E+2(4)	2.9E+1	4.02E+2(1)	3.2E+0
G_1_15_20_5.98	∗(2)	∗	∗(2)	∗	∗(2)	∗	7.02E+4(1)	1.8E+3
G_1_17_23_1.71	4.07E+2(1)	2.4E+1	6.10E+2(3)	2.5E+1	8.12E+2(4)	4.1E+1	4.58E+2(2)	3E+0
G_1_20_20_0.58	2.74E+2(2)	5.7E+0	2.88E+2(3)	2.5E+0	3.05E+2(4)	4.4E+0	2.60E+2(1)	3.2E+0
G_1_20_20_0.97	1.92E+2(2)	1.1E+1	2.53E+2(3)	6.3E+0	2.82E+2(4)	1.2E+1	1.45E+2(1)	2.3E+0
G_1_20_20_0.967	4.01E+2(1)	7.8E+0	4.26E+2(3)	5.7E+0	4.80E+2(4)	1.3E+1	4.08E+2(2)	4.6E+0
G_1_15_30_2.16	1.68E+3(2)	7.4E+1	2.09E+3(3)	5.8E+1	2.44E+3(4)	1.0E+2	1.41E+3(1)	2.2E+1
Rank value	61	-	66	-	99	-	30	-

The results of the Group 1 are shown in Tables 7 and 8 and Figures 3 and 4. MSGA is slightly worse than MA-MLS in G_1_15_20_0.67, G_1_17_23_1.71 and G_1_20_20_0.967; however, MSGA is better than the comparison algorithm in the remaining 24 instances in Table 7. In Table 8, MSGA provides worse results for 10 instances, but better results for the remaining 17 instances. For 27 benchmark instances on Group 1, the rank-order values of MSGA are 30 and 39, respectively. The difference of 1 between

MSGA's rank value and AC-ACO's rank value indicates that the two methods are not comparable in performance. Compared with other methods except for AC-ACO, the rank value of MSGA is the smallest.

Table 8. Comparison (II) of experimental results on Group 1.

Instance	ILS		AC-ACO		MSGA	
	Mean	Std	Mean	Std	Mean	Std
G_1_5_4_0.39	1.06E+2(2)	1.3E+0	1.06E+2(2)	1.3E+0	1.03E+2(1)	1.5E+0
G_1_5_5_1.66	1.20E+3(2)	5.7E+1	1.23E+3(3)	2.9E+1	1.13E+3(1)	2.1E+1
G_1_3_10_1.51	8.99E+2(3)	3.2E+1	8.77E+2(1)	1.5E+1	8.76E+2(1)	3.5E+0
G_1_3_15_5.03	4.00E+4(3)	7.4E+3	2.73E+4(1)	10.0E+1	2.76E+4(2)	8.3E+2
G_1_5_10_0.93	4.15E+2(2)	9.4E+0	4.09E+2(1)	8.2E+0	4.09E+2(1)	8.5E+0
G_1_10_5_1.39	6.01E+2(3)	3.6E+1	5.68E+2(2)	1.7E+1	5.65E+2(1)	2.1E+1
G_1_5_10_3.67	1.83E+4(3)	1.8E+3	1.31E+4(2)	7.1E+2	1.30E+4(1)	3.1E+3
G_1_5_20_4.36	3.35E+4(3)	4.7E+4	1.79E+4(2)	1.0E+3	2.03E+3(1)	4.3E+2
G_1_10_10_3.79	1.76E+4(3)	2.5E+3	7.44E+3(1)	3.7E+2	7.79E+3(2)	3.1E+2
G_1_11_11_1.28	3.36E+2(3)	2.3E+1	2.79E+2(2)	1.1E+1	2.50E+2(1)	2.0E+1
G_1_30_5_0.46	1.50E+2(1)	1.8E+0	1.50E+2(1)	1.9E+0	1.50E+2(1)	1.1E+0
G_1_15_10_1.17	3.61E+2(3)	2.1E+1	3.42E+2(2)	8.0E+0	3.17E+2(1)	1.2E+0
G_1_10_15_1.3	6.55E+2(3)	3.7E+1	6.30E+2(2)	1.3E+1	6.07E+2(1)	9.7E+0
G_1_20_10_0.47	9.85E+1(3)	1.8E+0	9.67E+1(1)	1.7E+0	9.77E+1(2)	1.8E+0
G_1_20_10_0.96	3.61E+2(3)	2.4E+1	3.41E+2(2)	7.2E+0	3.31E+2(1)	4.0E+0
G_1_20_10_0.94	2.70E+2(3)	1.6E+1	2.50E+2(2)	3.6E+0	2.39E+2(1)	4.5E+0
G_1_5_40_3.95	1.19E+4(2)	4.8E+2	1.17E+4(1)	6.3E+2	1.23E+4(3)	4.6E+2
G_1_10_20_6.04	$*$(3)	$*$	9.19E+4(2)	4.1E+3	9.10E+4(1)	5.12E+3
G_1_30_10_0.65	1.75E+2(3)	7.9E+0	1.61E+2(1)	2.9E+0	1.68E+2(2)	2.4E+0
G_1_15_20_0.67	3.54E+2(2)	1.0E+1	3.33E+2(1)	4.5E+0	3.60E+2(3)	3.1E+0
G_1_30_10_1.34	4.20E+2(3)	5.8E+1	3.68E+2(1)	8.5E+0	4.02E+2(2)	3.2E+0
G_1_15_20_5.98	$*$(3)	$*$	7.83E+4(2)	2.1E+3	7.02E+4(1)	1.8E+3
G_1_17_23_1.71	5.24E+2(3)	5.4E+1	3.43E+2(1)	1.5E+1	4.58E+2(2)	3E+0
G_1_20_20_0.58	2.73E+2(2)	5.5E+0	2.60E+2(1)	3.7E+0	2.60E+2(1)	3.2E+0
G_1_20_20_0.97	2.39E+2(3)	2.1E+1	1.65E+2(2)	6.3E+0	1.45E+2(1)	2.3E+0
G_1_20_20_0.967	4.35E+2(3)	3.1E+1	3.78E+2(1)	5.9E+0	4.08E+2(2)	4.6E+0
G_1_15_30_2.16	1.66E+3(3)	1.8E+2	1.34E+3(1)	3.6E+1	1.41E+3(2)	2.2E+1
Rank value	73	-	40	-	39	-

The computational results on Group 2 are shown in Tables 9 and 10 and Figures 5 and 6. In Table 9, MSGA's results are all top ranked. In Table 10, MSGA has worse results in G_2_40_10_0.67 and G_2_40_15_0.67, but better results in the remaining 4 examples. For 6 benchmark instances on Group 2, the rank-order of the MSGA are 6 and 9 , respectively. In Table 10, the rank value of AC-ACO is 10 and the rank value of MSGA is 9, which indicates that both methods perform similarly. Compared with other methods except for AC-ACO, MSGA has a better performance on Group 2.

The computational results on Group 3 are shown in Tables 11 and 12 and Figures 7 and 8. In

Table 11, the results of MSGA are all better than those of the comparison methods. Although in Table 12, MSGA is slightly worse than AC-ACO in G_3_60_30_1.14 and G_3_80_80_0.54, MSGA is better than the comparison methods in the other 15 instances. For 17 benchmark instances in Group 3, the rank-orders of the MSGA are 17 and 19, respectively. Compared with other algorithms, MSGA has a better performance in Group 3.

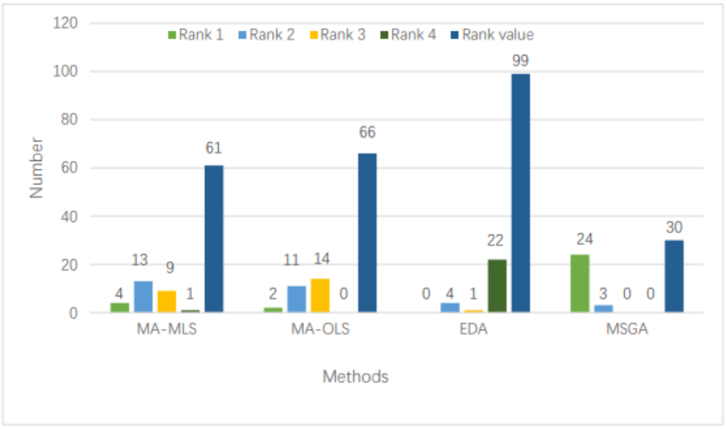


Figure 3. Ranking comparison (I) of experimental results on Group 1.

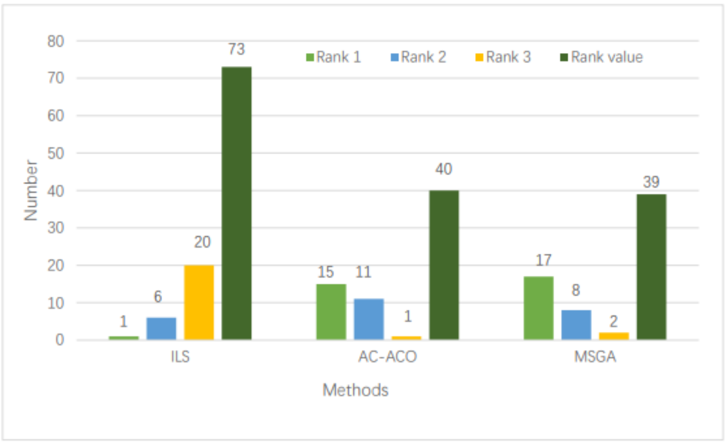


Figure 4. Ranking comparison (II) of experimental results on Group 1.

Table 9. Comparison (I) of experimental results on Group 2.

Instance	MA-MLS		MA-OLS		EDA		MSGA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
G_2_40_10_0.67	2.34E+2(2)	3.4E+0	2.39E+2(3)	1.3E+0	2.56E+2(4)	3.8E+0	2.29E+2(1)	3.2E+0
G_2_40_15_0.67	2.99E+2(1)	3.1E+0	3.05E+2(2)	2.2E+0	3.20E+2(3)	3.7E+0	2.98E+2(1)	2.5E+0
G_2_20_40_3.61	2.21E+4(2)	3.0E+3	5.28E+4(3)	4.6E+3	$*$(4)	$*$	5.35E+3(1)	3.2E+2
G_2_30_30_1.04	5.58E+2(2)	8.9E+0	5.80E+2(3)	6.7E+0	6.36E+2(4)	7.5E+0	4.30E+2(1)	2.4E+1
G_2_30_30_1.94	1.37E+3(2)	6.3E+1	1.54E+3(3)	4.3E+1	1.53E+3(3)	8.6E+1	8.78E+2(1)	3.2E+1
G_2_15_60_3.64	1.73E+4(2)	1.3E+3	2.50E+4(4)	1.2E+3	2.08E+4(3)	1.4E+3	5.71E+3(1)	3.1E+2
Rank value	11	-	18	-	21	-	6	-

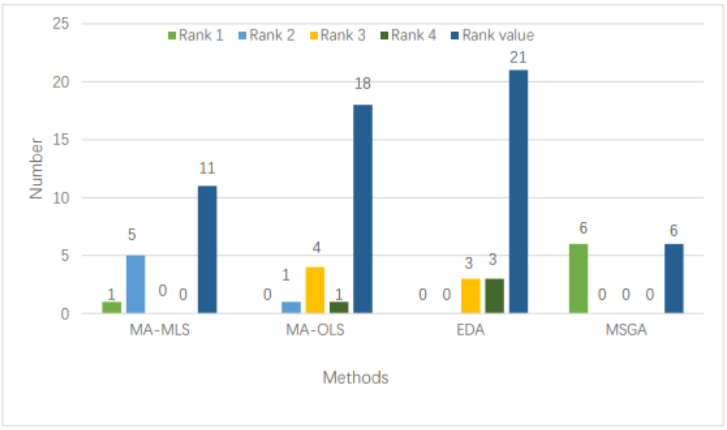


Figure 5. Ranking comparison (I) of experimental results on Group 2.

Table 10. Comparison (II) of experimental results on Group 2.

Instance	ILS		AC-ACO		MSGA	
	Mean	Std	Mean	Std	Mean	Std
G_2_40_10_0.67	2.37E+2(2)	7.2E+0	2.27E+2(1)	3.2E+0	2.29E+2(2)	3.2E+0
G_2_40_15_0.67	2.96E+2(2)	1.2E+1	2.80E+2(1)	4.3E+0	2.98E+2(3)	2.5E+0
G_2_20_40_3.61	1.32E+4(3)	4.0E+3	5.53E+3(2)	4.3E+2	5.35E+3(1)	3.2E+2
G_2_30_30_1.04	5.48E+2(3)	4.1E+1	4.66E+2(2)	6.4E+0	4.30E+2(1)	2.4E+1
G_2_30_30_1.94	1.31E+3(3)	7.8E+1	1.10E+3(2)	4.0E+1	8.78E+2(1)	3.2E+1
G_2_15_60_3.64	1.12E+4(3)	1.9E+3	1.01E+4(2)	4.8E+2	5.71E+3(1)	3.1E+2
Rank value	16	-	10	-	9	-

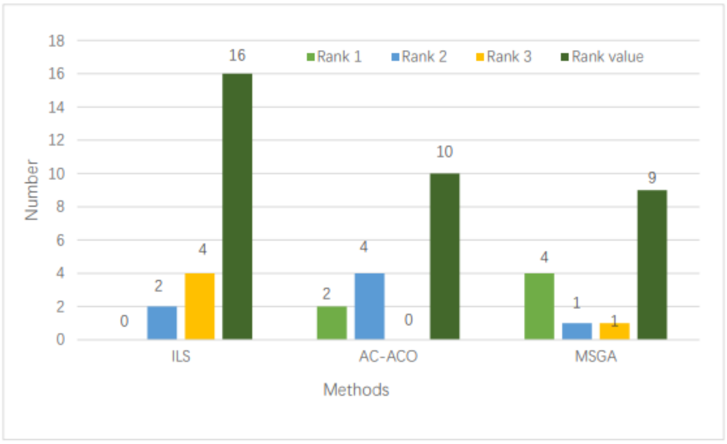


Figure 6. Ranking comparison (II) of experimental results on Group 2.

Because the experimental results of MSGA and AC-ACO are ranked, MSGA is ranked one place ahead of AC-ACO, so MSGA and AC-ACO perform about the same in solving Groups 1 and 2; however, there is a large ranking difference in Group 3 performance, with MSGA ranking 19 and AC-ACO ranking 32. Meanwhile, except for AC-ACO, MSGA has a smaller and larger ranking compared to all other methods. Therefore, it is not difficult to see that the MSGA is effective for solving these 50 benchmark instances.

Table 11. Comparison (I) of experimental results on Group 3.

Instance	MA-MLS		MA-OLS		EDA		MSGA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
G_3_80_15_0.76	1.98E+2(2)	2.3E+0	2.00E+2(3)	2.4E+0	2.07E+2(4)	3.5E+0	1.60E+2(1)	2.2E+0
G_3_20_60_1.51	1.01E+3(2)	2.9E+1	1.06E+3(3)	9.6E+0	1.15E+3(4)	1.4E+1	6.70E+2(1)	7.5E+0
G_3_80_20_0.7	3.18E+2(2)	3.1E+0	3.24E+2(3)	2.1E+0	3.35E+2(4)	4.1E+0	2.55E+2(1)	3.8E+0
G_3_80_20_1.12	3.90E+2(2)	6.2E+0	4.11E+2(3)	4.6E+0	4.53E+2(4)	1.1E+1	2.43E+2(1)	3.4E+0
G_3_20_80_4.33	1.25E+5(2)	2.5E+4	2.33E+5(3)	2.0E+4	$*$(4)	$*$	1.86E+4(1)	2.3E+2
G_3_60_30_1.14	5.52E+2(2)	9.9E+0	5.91E+2(3)	6.9E+0	6.95E+2(4)	2.2E+1	5.13E+2(1)	4.2E+0
G_3_60_40_0.69	3.91E+2(2)	3.4E+0	3.99E+2(3)	3.0E+0	4.04E+2(4)	4.4E+0	3.20E+2(1)	2.1E+0
G_3_40_60_1.54	9.48E+2(2)	1.7E+1	9.84E+2(3)	8.9E+0	1.05E+3(4)	1.5E+1	5.37E+2(1)	1.3E+1
G_3_80_40_0.97	4.64E+2(2)	5.4E+0	4.79E+2(3)	3.9E+0	5.01E+2(4)	8.5E+0	3.10E+2(1)	4.2E+0
G_3_40_80_1.05	6.42E+2(2)	8.1E+0	6.40E+2(3)	5.0E+0	6.60E+2(4)	9.5E+0	4.60E+2(1)	3.5E+0
G_3_80_40_2.25	6.54E+3(3)	1.0E+3	1.01E+4(4)	8.8E+2	3.69E+3(2)	3.4E+2	1.42E+3(1)	3.2E+1
G_3_60_60_0.92	4.29E+2(2)	5.3E+0	4.40E+2(4)	3.3E+0	4.36E+2(3)	6.2E+0	2.79E+2(1)	3.5E+0
G_3_60_60_0.922	5.51E+2(2)	6.3E+0	5.64E+2(3)	4.8E+0	5.56E+2(2)	5.8E+0	3.84E+2(1)	6.0E+0
G_3_120_30_1.2	4.44E+2(2)	6.1E+0	4.64E+2(3)	4.9E+0	4.94E+2(4)	1.2E+1	2.63E+2(1)	5.7E+0
G_3_80_60_0.72	4.18E+2(3)	4.1E+0	4.28E+2(4)	3.9E+0	4.17E+2(2)	4.7E+0	3.17E+2(1)	3.3E+0
G_3_80_80_0.54	3.23E+2(3)	3.3E+0	3.28E+2(4)	3.5E+0	3.08E+2(2)	2.5E+0	2.50E+2(1)	2.8E+0
G_3_60_120_2.07	3.25E+3(3)	6.5E+1	3.36E+3(4)	4.6E+1	3.09E+3(2)	8.2E+1	1.97E+3(1)	5.5E+1
Rank value	38	-	56	-	57	-	17	-

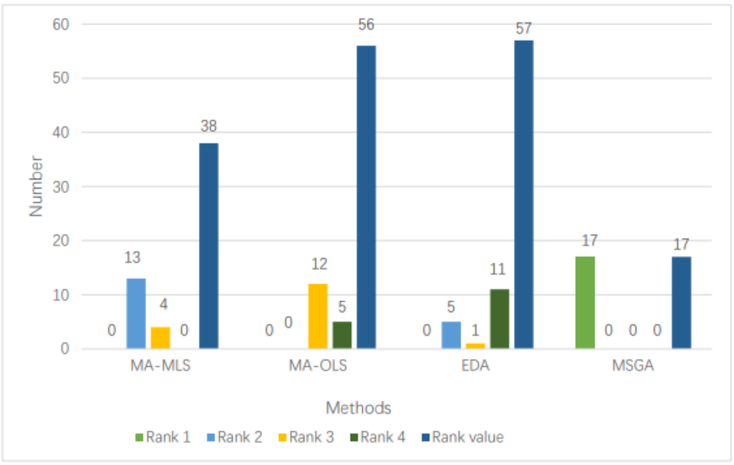


Figure 7. Ranking comparison (I) of experimental results on Group 3.

In addition, the average rankings of the six methods on Groups 1–3 and the 50 benchmark instances according to the Friedman's test are shown in Table 13, where MSGA is ranked less (1.59, 1.50, 1.12, 1.42) than the other methods. Although the gap between MSGA (1.50) ranking and AC-ACO (1.67) ranking is smaller on Group 2, there is a significant gap on Group 1, Group 3 and overall benchmark instances ranking, which means that MSGA performs better than others on overall benchmark instances.

Table 12. Comparison (II) of experimental results on Group 3.

Instance	ILS		AC-ACO		MSGA	
	Mean	Std	Mean	Std	Mean	Std
G_3_80_15_0.76	2.00E+2(3)	1.1E+1	1.70E+2(2)	3.6E+0	1.60E+2(1)	2.2E+0
G_3_20_60_1.51	9.60E+2(3)	2.7E+1	7.53E+2(2)	1.9E+1	6.70E+2(1)	7.5E+0
G_3_80_20_0.7	3.31E+2(3)	1.5E+1	2.86E+2(2)	5.5E+0	2.55E+2(1)	3.8E+0
G_3_80_20_1.12	4.16E+2(3)	2.2E+1	3.10E+2(2)	7.2E+0	2.43E+2(1)	3.4E+0
G_3_20_80_4.33	6.42E+4(3)	5.2E+3	2.21E+4(2)	2.0E+3	1.86E+4(1)	2.3E+2
G_3_60_30_1.14	6.21E+2(3)	3.7E+1	5.08E+2(1)	7.2E+0	5.13E+2(2)	4.2E+0
G_3_60_40_0.69	4.00E+2(3)	1.2E+1	3.40E+2(2)	3.0E+0	3.20E+2(1)	2.1E+0
G_3_40_60_1.54	9.05E+2(3)	2.3E+1	6.82E+2(2)	2.0E+1	5.37E+2(1)	1.3E+1
G_3_80_40_0.97	4.99E+2(3)	2.0E+1	3.86E+2(2)	5.2E+0	3.10E+2(1)	4.2E+0
G_3_40_80_1.05	6.23E+2(3)	1.7E+1	5.00E+2(2)	6.6E+0	4.60E+2(1)	3.5E+0
G_3_80_40_2.25	1.42E+4(3)	5.0E+3	1.69E+3(2)	6.6E+1	1.42E+3(1)	3.2E+1
G_3_60_60_0.92	4.48E+2(3)	1.8E+1	3.17E+2(2)	4.0E+0	2.79E+2(1)	3.5E+0
G_3_60_60_0.922	5.58E+2(3)	1.5E+1	4.18E+2(2)	6.9E+0	3.84E+2(1)	6.0E+0
G_3_120_30_1.2	4.97E+2(3)	1.8E+1	3.40E+2(2)	6.7E+0	2.63E+2(1)	5.7E+0
G_3_80_60_0.72	4.51E+2(3)	1.5E+1	3.29E+2(2)	2.8E+0	3.17E+2(1)	3.3E+0
G_3_80_80_0.54	3.54E+2(3)	1.5E+1	2.49E+2(1)	1.8E+0	2.50E+2(2)	2.8E+0
G_3_60_120_2.07	4.02E+3(3)	3.3E+2	2.19E+3(2)	6.9E+1	1.97E+3(1)	5.5E+1
Rank value	51	-	32	-	19	-

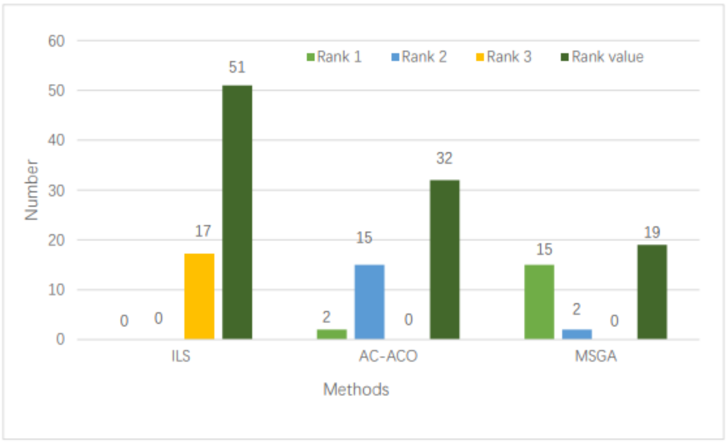


Figure 8. Ranking comparison (II) of experimental results on Group 3.

Results of the Wilcoxon signed-rank test on Group 1 in Table 14 provides larger R^+ values than R^- values in all cases, and the p-value is larger than 0.1 for AC-ACO. However, the p-values are less than 0.05 for MA-MLS, MA-OLS, EDA and ILS. This shows that the performance of MSGA is better than these methods.

Results of the Wilcoxon signed-rank test on Group 2 in Table 15 make clear in the same way that MSGA performed better than the other four methods. It provides larger R^+ values than R^- values in all cases, and the p-values are less than 0.05 for MA-MLS, MA-OLS, EDA and ILS, except for AC-ACO.

Results of the Wilcoxon signed-rank test on Group 3 in Table 16 show that MSGA performed better than the other five methods. It provides larger R^+ values than R^- values in all cases, and the p-values are less than 0.05.

So, overall, MSGA ranks first in the Friedman'test, results of the Wilcoxon signed-rank test on the 50 benchmark instances in Table 17 provide larger R^+ values than R^- values in all cases and the p-values are less than 0.05, showing that MSGA performs better than other methods.

Table 13. Rankings by the Friedman's test of MSGA and other methods.

Methods	Ranking (Group 1)	Ranking (Group 2)	Ranking (Group 3)	Ranking
MA-MLS	3.87	3.83	3.53	3.75
MA-OLS	4.20	5.33	4.62	4.48
EDA	5.74	5.67	5.00	5.48
ILS	3.69	3.00	4.85	4.00
AC-AC0	1.91	1.67	1.88	1.87
MSGA	1.59	1.50	1.12	1.42

Table 14. Results obtained by the Wilcoxon signed-rank test based on the mean values for MSGA and other methods on Group 1.

MSGA VS	R^+	R^-	p-value	$\alpha = 0.1$	$\alpha = 0.05$
MA-MLS	334	17	< 0.001	YES	YES
MA-OLS	351	0	< 0.001	YES	YES
EDA	378	0	< 0.001	YES	YES
ILS	331	20	< 0.001	YES	YES
AC-ACO	170.5	129.5	0.558	NO	NO

Table 15. Results obtained by the Wilcoxon signed-rank test based on the mean values for MSGA and other methods on Group 2.

MSGA VS	R^+	R^-	p-value	$\alpha = 0.1$	$\alpha = 0.05$
MA-MLS	21	0	0.028	YES	YES
MA-OLS	21	0	0.028	YES	YES
EDA	21	0	0.028	YES	YES
ILS	20	1	0.046	YES	YES
AC-ACO	18	3	0.116	NO	NO

On Groups 1 and 2, the performance of MSGA is comparable to that of AC-ACO, but better than the other four methods. On Group 3, the performance of MSGA is better than the performance of all five comparison methods. Overall, the performance of MSGA is better than the performance of the other five comparison methods in the 50 benchmark instances, indicating that MSGA is feasible and effective in solving MPDAP.

Table 16. Results obtained by the Wilcoxon signed-rank test based on the mean values for MSGA and other methods on Group 3.

MSGA VS	R^+	R^-	p-value	α = 0.1	α = 0.05
MA-MLS	153	0	< 0.001	YES	YES
MA-OLS	153	0	< 0.001	YES	YES
EDA	153	0	< 0.001	YES	YES
ILS	153	0	< 0.001	YES	YES
AC-ACO	150	3	< 0.001	YES	YES

Table 17. Results obtained by the Wilcoxon signed-rank test based on the mean values for MSGA and other methods on the 50 benchmark instances.

MSGA VS	R^+	R^-	p-value	α = 0.1	α = 0.05
MA-MLS	1202	23	< 0.001	YES	YES
MA-OLS	1225	0	< 0.001	YES	YES
EDA	1275	0	< 0.001	YES	YES
ILS	1186	39	< 0.001	YES	YES
AC-ACO	857.5	270.5	0.002	YES	YES

5.2. *Experimental results of MPDATP*

In this subsection, a simulation experiment is done based on the prioritization relationship of tasks. In a real environment, some tasks need to be prioritized according to their urgency due to their special location, which can lead to greater losses. In fact, when executing these tasks, it is not only necessary to consider the total completion time but, more importantly, to consider the urgency of the task. Therefore, priority should be given to the immediate task. For the convenience of the experiment, without loss of generality, 50 percent of a set of tasks are set to have priority constraints in the simulation experiment. The results of the simulation experiments are shown in Tables 18–20, where f refers to the objective function of the model.

According to the data in Tables 18–20, it can be seen that the time to complete a set of tasks increases after considering the task priority constraints set by the urgency degree of the tasks. However there is also a small number of instances in which the time to complete the task is the same as the completion time of the original model; this is because the randomly set task priority may be the same as the optimal solution for completing the task in the original model, which results in the same value of the objective function for the two models. The time to complete the tasks increases when there are task priority constraints, and in a practical setting it is possible to sacrifice the cost of time for the sake of life or some properties.

6. Conclusions

In the model of MPDATP, a set of tasks needs to be prioritized. In this manuscript, the tasks prioritization relationships were considered based on the urgency of the task. In order to efficiently

Table 18. The solutions of prioritized relational model for Group 1.

Instance	Best solution of prioritized relational model	Best solution of original model
	f	f
G_1_5_4_0.39	1.03E+2	1.03E+2
G_1_5_5_1.66	1.16E+3	1.13E+3
G_1_3_10_1.51	1.10E+3	8.76E+2
G_1_3_15_5.03	2.76E+4	2.75E+4
G_1_5_10_0.93	6.23E+2	4.09E+2
G_1_10_5_1.39	6.02E+2	5.65E+2
G_1_5_10_3.67	1.34E+4	1.30E+4
G_1_5_20_4.36	2.03E+3	2.03E+3
G_1_10_10_3.79	8.56E+3	7.79E+3
G_1_11_11_1.28	3.71E+2	2.50E+2
G_1_30_5_0.46	2.68E+2	1.50E+2
G_1_15_10_1.17	7.16E+2	3.17E+2
G_1_10_15_1.3	6.50E+2	6.07E+2
G_1_20_10_0.47	3.01E+2	9.77E+1
G_1_20_10_0.96	5.23E+2	3.31E+2
G_1_20_10_0.94	4.47E+2	2.39E+2
G_1_5_40_3.95	3.58E+4	1.23E+4
G_1_10_20_6.04	1.17E+5	9.10E+4
G_1_30_10_0.65	4.46E+2	1.68E+2
G_1_15_20_0.67	6.61E+2	3.60E+2
G_1_30_10_1.34	7.37E+2	4.02E+2
G_1_15_20_5.98	1.55E+5	7.02E+4
G_1_17_23_1.71	4.58E+2	4.58E+2
G_1_20_20_0.58	5.27E+2	2.60E+2
G_1_20_20_0.97	1.45E+2	1.45E+2
G_1_20_20_0.967	4.95E+2	4.08E+2
G_1_15_30_2.16	2.11E+3	1.41E+3

Table 19. The solutions of bilevel programming model for Group 2.

Instance	Best solution of prioritized relational model	Best solution of original model
	f	f
G_2_40_10_0.67	3.07E+2	2.29E+2
G_2_40_15_0.67	4.97E+2	2.98E+2
G_2_20_40_3.61	6.60E+3	5.35E+3
G_2_30_30_1.04	9.09E+2	4.30E+2
G_2_30_30_1.94	1.42E+3	8.78E+2
G_2_15_60_3.64	5.80E+3	5.71E+3

Table 20. The solutions of bilevel programming model for Group 3.

Instance	Best solution of prioritized relational model	Best solution of original model
	f	f
G_3_80_15_0.76	4.96E+2	1.60E+2
G_3_20_60_1.51	5.55E+3	6.70E+2
G_3_80_20_0.7	5.13E+2	2.55E+2
G_3_80_20_1.12	6.49E+2	2.43E+2
G_3_20_80_4.33	1.26E+5	1.86E+4
G_3_60_30_1.14	6.15E+2	5.13E+2
G_3_60_40_0.69	5.99E+2	3.20E+2
G_3_40_60_1.54	1.57E+3	5.37E+2
G_3_80_40_0.97	9.68E+2	3.10E+2
G_3_40_80_1.05	8.13E+2	4.60E+2
G_3_80_40_2.25	1.15E+4	1.42E+3
G_3_60_60_0.92	7.67E+2	2.79E+2
G_3_60_60_0.922	1.16E+3	3.84E+2
G_3_120_30_1.2	8.10E+2	2.63E+2
G_3_80_60_0.72	1.08E+3	3.17E+2
G_3_80_80_0.54	1.00E+3	2.50E+2
G_3_60_120_2.07	3.21E+3	1.97E+3

solve this problem, an MSGA was proposed, in which a two-stage hybrid matrix encoding was given to show the route of each robot for the tasks with priority constraints. Meanwhile, to improve the diversity and feasibility of individuals, three crossover operators based on matrix operations and two individual repair operators based on rationally adjusting the inverse order were given, respectively. From the experimental results, the proposed MSGA solving MPDAP was effective. Moreover, the MSGA also provided decision makers with effective task completion options in cases where some tasks have priority constraints.

In the future, there are still some interesting topics that need to be further investigated. There is also a need to design optimization operators and optimization learning techniques that are more suitable for solving such practical problems, as well as to provide decision makers with more sound theoretical guidance.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the National Science Foundation of China (No.61966030).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. B. Xin, Y. G. Zhu, Y. L. Ding, G. Q. Gao, Coordinated motion planning of multiple robots in multi-point dynamic aggregation task, in *2016 12th IEEE International Conference on Control and Automation (ICCA)*, Kathmandu, Nepal, (2016), 933–938. https://doi.org/10.1107/ICCA.2016.7505398
2. Y. L. Liao, K. L. Su, Multi-robot-based intelligent security system, *Artif. Life Rob.*, **16** (2011), 137. https://doi.org/10.1007/s10015-011-0888-x
3. C. Y. Ju, J. Kim, J. Seol, H. I. Son, A review on multirobot systems in agriculture, *Comput. Electron. Agric.*, **202** (2022), 107336. https://doi.org/10.1016/j.compag.2022.107336
4. V. Akbari, F. S. Salman, Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity, *Eur. J. Oper. Res.*, **257** (2017), 625–640. https://doi.org/10.1016/j.ejor.2016.07.043
5. W. Q. Jin, S. Q. Dong, C. Q. Yu, Q. Q. Luo, A data-driven hybrid ensemble AI model for COVID-19 infection forecast using multiple neural networks and reinforced learning, *Comput. Biol. Med.*, **146** (2022), 105560. https://doi.org/10.1016/j.compbiomed.2022.105560
6. N. N. Zheng, S. Y. Du, J. J. Wang, H. Zhang, W. T. Cui, Z. J. Kang, et al., Predicting COVID-19 in China using hybrid AI model, *IEEE Trans. Cybern.*, **50** (2020), 2891–2904. https://doi.org/10.1109/tcyb.2020.2990162
7. C. Robin, S. Lacroix, Multi-robot target detection and tracking: taxonomy and survey, *Auton. Rob.*, **40** (2016), 729–760.
8. R. P. Yuan, J. T. Li, X. L. Wang, L. Y. He, Multirobot task allocation in e-Commerce robotic mobile fulfillment systems, *Math. Probl. Eng.*, **2021** (2021), 6308950. https://doi.org/10.1155/2021/6308950
9. A. Khan, B. Rinner, A. Cavallaro, Cooperative robots to observe moving targets: Review, *IEEE Trans. Cybern.*, **48** (2018), 187–198. https://doi.org/10.1109/TCYB.2016.2628161
10. G. A. Korash, A. Stentz, M. Dias, A comprehensive taxonomy for multi-robot task allocation, *Int. J. Rob. Res.*, **32** (2013), 1495–1512. https://doi.org/10.1177/0278364913496484
11. B. P. Gerkey, M. J. Mataric, A formal analysis and taxonomy of task allocation in multi-robot systems, *Int. J. Rob. Res.*, **23** (2004), 939–954. https://doi.org/10.1177/0278364904045564
12. G. Q. Gao, Y. Mei, Y. H. Jia, W. N. Browne, B. Xin, Adaptive coordination ant colony optimization for multipoint dynamic aggregation, *IEEE Trans. Cybern.*, **52** (2022), 7362–7376. https://doi.org/10.1109/TCYB.2020.3042511
13. B. Xin, S. Liu, Z. Peng, G. Gao, An estimation of distribution algorithm for multi-robot multi-point dynamic aggregation problem, in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Miyazaki, Japan, (2018), 775–780.

14. S. Lu, B. Xin, L. Dou, L. Wang, A multi-model estimation of distribution algorithm for agent routing problem in multi-point dynamic task, in *Proceedings of the 37th Chinese Control Conference (CCC)*, Wuhan, China, (2018), 2468–2473.

15. G. Q. Gao, Y. Mei, X. Bin, Y. H. Jia, W. N. Browne, A memetic algorithm for the task allocation problem on multi-robot multi-point dynamic aggregation missions, in *2020 IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, U.K., (2020), 1–8.

16. R. Hao, J. Zhang, B. Xin, C. Chen, L. Dou, A hybrid differential evolution and estimation of distribution algorithm for the multi-point dynamic aggregation problem, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, (2018), 251–252.

17. J. Chen, Y. Guo, Z. Qiu, B. Xin, Q. S. Jia, W. H. Gui, Multiagent dynamic task assignment based on forest fire point model, *IEEE Trans. Autom. Sci. Eng.*, **19** (2022), 833–849. https://doi.org/10.1109/TASE.2021.3061757

18. R. P. Yuan, J. T. Dou, J. T. Li, W. Wang, Y. F. Jiang, Multi-robot task allocation in e-commerce RMFS based on deep reinforcement learning, *Math. Biosci. Eng.*, **20** (2023), 1903–1918. https://doi.org/10.3934/mbe.2023087

19. X. B. Zhou, X. Cai, H. Zhang, Z. H. Zhang, T. Jin, H. Y. Chen, et al., Multi-strategy competitive-cooperative co-evolutionary algorithm and its application, *Inf. Sci.*, **635** (2023), 328–344. https://doi.org/10.1016/j.ins.2023.03.142

20. C. Huang, X. Zhou, X. Ran, J. Wang, H. Chen, W. Deng, Adaptive cylinder vetor particle syarm optimization with differential evolution for UAV path planning, *Eng. Appl. Artif. Intell.*, **121** (2023), 105942. https://doi.org/10.1016/jengappai.2023.105942

21. X. Wu, J. Peng, Z. Xie, N. Zhao, S. Wu, An improved muli-obiective optimization algorithm for solving flexible job shop scheduling problem with variable batches, *J. Syst. Eng. Electron.*, **32** (2021), 272–285. https://doi.org/10.23919/JSEE.2021.000024

22. J. Xu, Y. L. Zhao, H. Y. Chen, W. Deng, ABC-GSPBFT: PBFT with grouping score mechanism and optimized consensus pro-cess for flight operation data-sharing, *Inf. Sci.*, **624** (2023), 110–127. https://doi.org/10.1016/j.ins.2022.12.068

23. M. Li, J. Y. Zhang, J. Song, Z. J. Li, S. F. Lu, A clinical-oriented non-Severe depression diagnosis method based on cognitive behavior of emotional conflict, *IEEE Trans. Comput. Social Syst.*, **10** (2023), 131–141. http://dx.doi.org/10.1109/TCSS.2022.3152091

24. M. Li, W. Zhang, B. Hu, J. M. Kang, Y. Q. Wang, S. F. Lu, Automatic assessment of depression and anxiety through encoding pupil-wave from HCI in VR scenes, *ACM Trans. Multimedia Comput. Commun. Appl.*, **20** (2023), 1–22. https://doi.org/10.1145/3513263

25. Z. Yan, H. Y. Yang, Y. K. Wu, Y. Lin, A multi-view attention-based spatia Ctemporal network for airport arrival flow prediction, *Transp. Res. Part E Logist. Transp. Rev.*, **170** (2023), 102997. https://doi.org/10.1016/j.tre.2022.102997

26. H. J. Wang, Z. J. Fu, J. J. Zhou, M. Y. Fu, R. Li, Cooperative collision avoidance for unmanned surface vehicles based on improved genetic algorithm, *Ocean Eng.*, **222** (2021), 108612. https://doi.org/10.1016/j.oceaneng.2021.108612

27. N. Milad, K. Esmaeel, D. Samira, Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm, *Expert Syst. Appl.*, **115** (2019), 106–120. https://doi.org/10.1016/j.eswa.2018.08.008
28. J. F. Xin, J. B. Zhong, F. R. Yang, Y. Cui, J. L. Sheng, An improved genetic algorithm for path-planning of unmanned surface vehicle, *Sensors*, **19** (2019), 2640. https://doi.org/10.3390/s19112640
29. X. H. Guo, M. J. Ji, W. D. Zhang, Research on a new two-level scheduling approach for unmanned surface vehicles transportation containers in automated terminals, *Comput. Ind. Eng.*, **175** (2023), 108901. https://doi.org/10.1016/j.cie.2022.108901
30. G. Xia, X. Sun, X. Xia, Multiple task assignment and path planning of a multiple unmanned surface vehicles system based on improved self-organizing mapping and improved genetic algorithm, *J. Mar. Sci. Eng.*, **9** (2021), 556. https://doi.org/10.3390/jmse9060556
31. H. Y. Lee, H. Shin, J. Chae, Path planning for mobile agents using a genetic algorithm with a direction guided factor, *Electronics*, **7** (2018), 212. https://doi.org/10.3390/electronics7100212
32. A. K. Cechinel, E. R. D. Pieri, A. L. F. Perez, P. D. M. Plentz, Multi-robot task allocation using island model genetic algorithm, *IFAC-PapersOnLine*, **54** (2021), 558–563. https://doi.org/10.1016/j.ifacol.2021.08.063
33. Z. Entezari, M. Mahootchi, Developing a mathematical model for staff routing and scheduling in home health care industries: genetic algorithm-based solution scheme, *Sci. Iran.*, **28** (2021), 3692–3718. https://doi.org/10.24200/SCI.2020.54116.3600
34. C. Ramirez-Atencia, G. Bello-Orgaz, M. D. R-Moreno, D. Camacho, Solving complex multi-UAV mission planning problme using multi-objective genetic algorithms, *Soft Comput.*, **21** (2017), 4883–4900. https://doi.org/10.1007/s00500-016-2376-7
35. X. Wang, T. M. Choi, Z. Li, S. Shao, An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem, *IEEE Trans. Syst. Man Cybern. Syst.*, **50** (2020), 4948–4958. https://doi.org/10.1109/TSMC.2019.2938298