Mathematical Biosciences and Engineering

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

MBE, 20(8): 13754–13776.
DOI: 10.3934/mbe.2023612
Received: 06 April 2023
Revised: 16 May 2023
Accepted: 29 May 2023
Published: 16 June 2023

Research article

A new differential evolution using a bilevel optimization model for solving generalized multi-point dynamic aggregation problems

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

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

[2] School of Mathematics and Statistics, Qinghai Normal University, Xining 810008, Qinghai, China

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

Abstract: The multi-point dynamic aggregation problem (MPDAP) comes mainly from real-world applications, which is characterized by dynamic task assignation and routing optimization with limited resources. Due to the dynamic allocation of tasks, more than one optimization objective, limited resources, and other factors involved, the computational complexity of both route programming and resource allocation optimization is a growing problem. In this manuscript, a task scheduling problem of fire-fighting robots is investigated and solved, and serves as a representative multi-point dynamic aggregation problem. First, in terms of two optimized objectives, the cost and completion time, a new bilevel programming model is presented, in which the task cost is taken as the leader's objective. In addition, in order to effectively solve the bilevel model, a differential evolution is developed based on a new matrix coding scheme. Moreover, some percentage of high-quality solutions are applied in mutation and selection operations, which helps to generate potentially better solutions and keep them into the next generation of population. Finally, the experimental results show that the proposed algorithm is feasible and effective in dealing with the multi-point dynamic aggregation problem.

Keywords: multi-point dynamic aggregation problem; multirobot system; task allocation; bilevel optimization model; differential evolution

1. Introduction

The multi-point dynamic aggregation problem (MPDAP) [1] is widely used in resource allocation and robot routing scheduling problems, such as post-disaster rescue [2], e-commerce logistics, forest fire fighting [3], medical equipment scheduling [4], and other task allocation optimization problems [5, 6]. As a representative of the problems, robot routing optimization with dynamic task scheduling is an interesting research issue in intelligent optimization community. For the purpose of improving the

efficiency of completing tasks, the moving routes of the robots may often be adjusted according to the observed tasks. Another characteristic of the problem is that there are multiple robots involved, that is to say, this is a multi-robot task scheduling problem. Some interesting real-world applications, such as e-commerce robots [5] and agricultural robots [6], have been presented in which the multi-robot task allocation models are developed and solved. In these problems, one needs to consider that each may change over time, the path of each robot, and the cooperation with other robots.

To better illustrate the problem, a two-robot and five-task case is present in Figure 1 and Figure 2. Figure 1 is a schematic diagram of two robots cooperating to execute five tasks according to their own driving path from the depot, in which the rectangle is the depot, the black circle represents the robot, and the white circle stands for the task. The solid line is the driving path of robot 1, and the dashed line is the driving path of robot 2. Figure 2 shows the time-demand change diagram of task 3 which needs to be executed by the robots. In Figure 2, point A gives the initial state value of task 3, point B provides the total task demand facing robot 1 and robot 2 when they reach task point 3, and the task demand at point C approaches 0, indicating that task 3 has been cooperatively completed by both robot 1 and robot 2. It can also be seen from Figure 2 that the cumulative time from the depot to completing task 3 is 5.3. One methodology to simplify the problem is to minimize the time cost at which all five tasks are completed by the two robots.

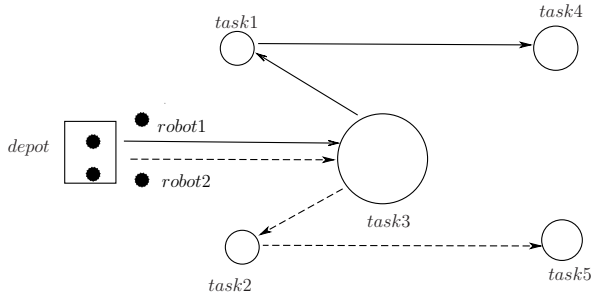

Figure 1. Task allocation problem.

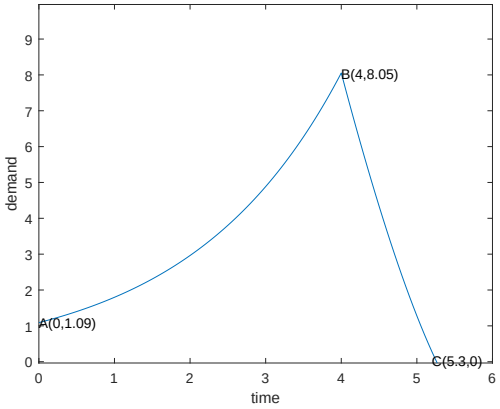

Figure 2. Time-demand curve for task3 in the figure 1.

The above-mentioned multi-point dynamic aggregation problem is a combination of path optimization problems and the task assignment of multi-robot systems. It not only needs to plan the path of each robot to execute the task, but also reasonably adjust the cooperation between robots to complete the tasks at which the time-demanding amount may be dynamically changing. Therefore, the multi-point dynamic aggregation problem has the following challenges. First, in order to complete a task requires the cooperation of robots, each robot needs to consider its own path as well as plans of other robots during the task execution. Second, the time each robot requires to complete a certain task depends on the growth rate of task demand at each task, which is a dynamically decision-making procedure. Third, the optimization of both the traveling routes of robots and other objectives has to be made in a multi-factor interactive environment involving dynamic task allocation and cooperation among multiple individuals, etc.

Despite the computational complexity of the problem, the wide range of practical applications has stimulated the study of modeling and solution methods of the problem. In [7], a model of forest fire suppression was presented as a multiple permutation combinatorial optimization problem, named MPDAP, and a distribution estimation algorithm using an effective encoding method was proposed for this problem. The given experimental show that the proposed algorithm can find a better solution for MPDAP than other compared approaches. In [8], a nonlinear agent routing problem in multi-point dynamic task (ARP-MPDT) was developed, which was based on a node histogram model (NHM), an edge histogram model (EHM) and probabilistic modeling distribution estimation algorithm (EDA), in which the ratio of NHM and EHM probability models can be adjusted adaptively. Gao et al. [9] represented each robot task by multiple row access sequence coding which designed a kind of heuristic rule of implicit decoding strategy to simplify the problem from the mixed-discrete variable optimization, and was based on a one-step and multi-step local search and presented a memetic algorithm for solving the problem. Hao et al. [10] combined the advantages of differential evolution algorithms and distribution estimation algorithms, and then proposed a hybrid algorithm. The proposed algorithm was executed on MPDAP examples of different sizes, and the simulation results show that compared with single differential evolution or distribution estimation algorithms, the hybrid method can solve this problem more efficiently. In an ant colony framework, in order to improve the efficiency of the solution construction, a new pheromone matrix and pheromone updating mechanism were proposed [11]. In order to reduce the search space and delete some bad solutions, Gao et al. [11] designed a heuristic value evaluation function in solution construction, and a pheromone repair based mechanism was investigated to enhance the ability to build viable solutions, and a local search was adopted to improve the balance between exploration and exploitation.

It should be noted that most of existing models for MPDAPs work to minimize the completion time of all tasks, in which the number of robots is predetermined. A recurring question is "how may robots are best?", that is to say, from the point of view of controlling total costs involving robots, time costs and material damage, how does a decision-maker optimize the number of dispatching robots and the traveling lines of them? Different from the above literatures, in the manuscript, a new bilevel programming problem is provided and solved by designing a new encoding scheme and efficient evolutionary operators. Some main innovations of this manuscript are presented as follows:

1) In the proposed model, the total cost of performing all tasks, instead of only time cost, is taken into account, in which the number of robots need to be optimized. Hence, the existing model with a predetermined number of robots is simply a special case of the proposed model. In the proposed model, both the robot cost and completion time are located in different levels and hierarchically optimized.

2) A new matrix encoding scheme is developed for all follower's solutions, which is simpler in form and more efficient than most of the existing encoding methods.

3) Both novel two-strategy mutation and selection schemes are designed and adopted in different evolution stages, which helps to find better potential solutions.

The rest of the manuscript is arranged as follows. Section 2 introduces bilevel models and differential evolution methods. Section 3 introduces the MPDAP model. The detailed algorithm design is presented in Section 4. The experimental studies are implemented in Section 5. Section 6 provides a discussion. Finally, Section 7 concludes this manuscript.

2. Preliminaries

2.1. Bilevel optimization models

Bilevel programming is a hierarchical optimization problem, in which variables are divided and located at two levels, the leader's and follower's levels, respectively. A general bilevel optimization model can be formulated as

$$\begin{cases} \min\limits_{x\in X} F(x,y) \\ s.t \quad G(x,y) \le 0 \\ \min\limits_{y\in Y} f(x,y) \\ s.t \quad \mathrm{g}(x,y) \le 0 \end{cases} \tag{2.1}$$

where

$$\begin{cases} \min\limits_{x\in X} F(x,y) \\ s.t \quad G(x,y) \le 0 \end{cases} \tag{2.2}$$

and

$$\begin{cases} \min\limits_{x\in Y} f(x,y) \\ s.t \quad \mathrm{g}(x,y) \le 0 \end{cases} \tag{2.3}$$

are called the leader's and follower's problems, respectively. $F(x,y)$ and $f(x,y)$ are the leader's and follower's objective functions, respectively, $G(x,y)$ and $g(x,y)$ are the constraints of the leader's and follower's problems, respectively, and , x and y are called the leader's and follower's variables, respectively. The bilevel programming problem is widely used in many practical fields, such as economics, engineering science, operation research, and optimal control [12], and has a decision-making system with a hierarchical (nested) optimization process, in which multiple decision-makers with different priorities locates at different decision-making levels. Generally speaking, the priority of decision-makers at the higher level is higher than that of decision-makers at the lower level. The decision-makers at the higher level regulate the decision-making process at the lower level through optimization procedures, and the decision of the follower can also affect the optimization result of the leader's problem.

The computational difficulty of solving the bilevel optimization problem lies in the large amount of computation caused by frequently solving the follower's problems. At present, the methods to solve bilevel optimization problems mainly include the traditional optimization method using gradient descent, swarm intelligent approaches, and hybrid technologies. The traditional optimization methods include the single level reduction method, the branch and bound method, the penalty function method, the trust region method, and so on. Sinha et al. [13] used Karush-Kuhn-Tucher (K-K-T) conditions to transform the bilevel optimization model into a single-level optimization problem, though the transformed model always causes a complex constraint restriction. For integer bilevel optimization problems, Liu et al. [14] proposed a branch and bound method based on a relax and check procedure. Li et al. [15] proposed a numerical method for solving bilevel optimization problems with non-convex follower's problems, in which, the follower's problem is replaced by an equivalent substitution through K-K-T conditions. Joseph et al. [16] proposed a bilevel optimization model for classification feature

selection problems and designed a genetic algorithm based on derivatively-free optimization. Li et al. [17] studied a class of linear fraction bilevel optimization problems where the coefficients and the right vector of the objective function are interval numbers and proposed an evolutionary algorithm based on optimality conditions. Aboelnage et al. [18] proposed an improved algorithm based on a new selection method and a chaotic search method for bilevel optimization problems. Goshu et al. [19] proposed a meta-heuristic algorithm based on a random space system sampling technique to solve the stochastic bilevel optimization problem. The hybrid optimization method combines the advantages of various algorithms to deal with the bilevel optimization problem, and shows a good performance. Islam et al. [20] proposed a memetic algorithm combining global search and local search. Yousria et al. [21] proposed a hybrid algorithm combining an improved genetic algorithm and chaotic search techniques. Sinha et al. [22] proposed a bilevel optimization algorithm with a multi-valued iterative approximation, which uses the follower's reaction set mapping to reduce the computation. Molai [23] used an extension principle for fuzzy multi-objective linear bi-level optimization problems.

Most of the existing methods are efficient in solving bilevel programming problems with small scale convex and differentiable functions. However, when the scale of the problem is sharply increased and non-convex, either non-differential functions or discrete variables are involved and the performance of these approaches always degenerates rapidly and can't be used to deal with the problem of this kind.

2.2. Differential evolution

Differential evolution (DE) is a very popular evolutionary algorithm paradigm proposed by Storn and Price [24]. It contains four processes: initialization, mutation, crossover, and selection.

Initialization: There are N individuals generated randomly in a feasible region.

$$P(T) = \{X_1, X_2, \cdots, X_m, \cdots, X_N\} \tag{2.4}$$

where X_m is the mth individual.

Mutation: The mutation operation is the main way for populations to generate new individuals, which can increase population diversity. For each individual X_m, a mutation vector V_m is generated. Some common mutation operators are as follows

DE/rand/1/bin:

$$V_m = X_{r_1} + F_{sf}(X_{r_2} - X_{r_3}) \tag{2.5}$$

DE/best/1/bin:

$$V_m = X_{best} + F_{sf}(X_{r_1} - X_{r_2}) \tag{2.6}$$

DE/rand/2/bin:

$$V_m = X_{r_1} + F_{sf_1}(X_{r_2} - X_{r_3}) + F_{sf_2}(X_{r_4} - X_{r_5}) \tag{2.7}$$

DE/best/2/bin:

$$V_m = X_{best} + F_{sf_1}(X_{r_2} - X_{r_3}) + F_{sf_2}(X_{r_4} - X_{r_5}) \tag{2.8}$$

DE/rand-to-best/1/bin:

$$V_m = X_{r_1} + rand(0, 1)(X_{best} - X_{r_1}) + F_{sf_2}(X_{r_2} - X_{r_3}) \tag{2.9}$$

where r_1, r_2, r_3, r_4, and r_5 are five different integers chosen randomly from $[1, N]$, X_{best} is the best individual in the population, $rand(0, 1)$ is a uniformly distributed random number from $[0, 1]$, and F_{sf}, F_{sf_1}, and F_{sf_2} are scaling factors.

Crossover: Using binomial crossover to generate crossover offspring U_m.

Selection: The better one between X_m and U_m is selected into the next generation.

DE uses the differential information of multiple individuals in a population to achieve perturbation of evolved individuals. As s one of the effective swarm intelligent algorithms, DE has been successfully applied to many practical problems [25]. Wang et al. [26] used a new encoding method based differential evolution to solve the distribution problem of wind farms. Aiming at power optimization problem of power system, Chi et al. [27] adopted a differential evolution algorithm based on different evolutionary operations. Additionally, other interesting applications can be found, such as economic or emission dispatch problem [28], image segmentation [29], and task scheduling in cloud computing environment [30]. As a bilevel programming model with discrete variables, the proposed MPDAP in the manuscript is very hard to solve. In view of the good performance of differential evolution in dealing with complex optimization problems, this paper developed an improved version of this algorithm to solve the proposed MPDAP model.

3. Multi-point dynamic aggregation problem

3.1. Demand state model for task in MPDAP

From the introduction, it can be seen that in a MPDAP, the task demand of each task (e.g. fire point) may dynamically increase over time. The relationship between task demand and time is suggested by the following equations in [11]:

$$q_i^t = q_i^0 + \alpha_i t \tag{3.1}$$

where, q_i^t represents the demand at task i at time t, the initial demand of task i is q_i^0, and α_i is the inherent increment rate of task i.

3.2. The model of the MPDAP

The workload of the robot r to solve the task i can be expressed by the following formula:

$$t_{task_i} = \frac{S_{task_i}}{v_{task}^r} \tag{3.2}$$

where, S_{task_i} represents the total amount of work that robot does, v_{task}^r is the work efficiency, and t_{task_i} stands for the time.

$S_{path_{ji}}$ is the path length from task j to task i, the speed of the robot r is v_{path}^r, and the time taken from task j to task i is $t_{path_{ji}}$. The formula is as follows:

$$t_{path_{ji}} = \frac{S_{path_{ji}}}{v_{path}^r} \tag{3.3}$$

In [11], the objective function in the MPDAP model is

$$\min f = \max_{i=1,2,\cdots,M_1} t_i \tag{3.4}$$

where, t_i is the completion time of task i. In fact, in a real-world planing problem, it is very important to control overall cost, not just time cost. In this manuscript, a more general case is taken into account,

which involves the cost of operation time, damage caused by fire, and robot cost. In the model, once the number of robots are determined, one can program the route and assign a task for each robot. In this procedure, a hierarchical optimization process arises. As a result, a bilevel programming model is developed as follows:

$$\begin{cases} \min F = F_1 + F_2 + F_3 \\ \min f = \underset{i=1,2,\cdots,M_1}{max} t_i \end{cases} \tag{3.5}$$

where, the leader's objective function F stands for the cost of all the robots to complete the tasks and the amount of damage in the tasks; the follower's objection function f is the maximum completion time of all the tasks.

$$F_1 = \sum_{i}^{M_1} \lambda_1 q_i^{t_i} \tag{3.6}$$

$$F_2 = \sum_{r}^{M_2} c_r^0 \tag{3.7}$$

$$F_3 = \sum_{r}^{M_2} c_r (min \underset{i=1,2,\cdots,M_1}{max} t_i) \tag{3.8}$$

$$t_i = t_j + t_{path_{ji}} + t_{task_i} \tag{3.9}$$

Equation (3.6) represents the total amount of damages at all M_1 tasks when all tasks are completed, in which the total amount of damage at task i is λ_1 times the task demand at task i. Equation (3.7) is the state-determined cost of renting (or buying) M_2 robots used in the tasks, here, c_r^0 is the inherent cost of robot r. Equation (3.8) is the running cost of all the robots and c_r is the unit running cost of robot r.

As is known, model (3.4) is simply the follower's problem of model (3.5). The number of robots and inherent costs in model (3.5) are fixed, which is degenerated to model (3.4). Therefore, model (3.5) is more general and extensive than model (3.4).

In order to make the model of multiple robots and tasks more explicit, relevant assumptions in [11] are also adopted in this manuscript:

1) There are M_2 robots, and their initial location is the depot. The cost of robots of the same size is the same, and they have the same traveling speed v_{path} and the work efficiency v_{task}.

2) There are M_1 tasks in different positions, the tasks at different locations may have the same or different task demand.

3) In the working environment, there are no obstacles, and each robot does not have to consider the collision with others. Once the driving path is determined, the robot can drive.

4) The starting time that the robots leaves the depot is denoted as 0.

5) For each task, the number of ways into the task point equals to that of ways out.

6) Each task is performed by at least one robot.

7) Each task is executed by a robot at most once.

8) Once the current task is completed, the robot can move on to the next.

9) The moving path of each robot can be planned in advance before performing tasks.

4. Algorithmic design

In the proposed bilevel model (3.5), the leader's problem is to optimize the number of robots assigned to tasks. Once the number of the robots is determined, the follower needs to optimize the moving routes of robots. Since the optimization procedure is hierarchical and most of the computational amounts mainly come from the solutions of the follower's problem, the manuscript is first focused on the follower's algorithm design. Finally, by embedding the follower's algorithm, a novel bilevel differential evolution (NDE) is presented for the bilevel model (3.5).

4.1. Algorithmic design of the follower's problem

This section mainly introduces the main schemes, as well as the follower's algorithm. First of all, a new encoding scheme is developed, which can more efficiently express the robot's traveling path and completion tasks than others. In addition, a two-strategy mutation is provided by using the mean of locations of the top individuals, which can efficiently avoid the directional mislead from a single good point.

4.1.1. Encoding / Decoding strategies

In an MPDAP tasks planning problem with M_1 tasks and M_2 robots, a feasible solution is encoded as an $M_2 \times (M_1 + 1)$ matrix X:

$$X_m = \begin{bmatrix} 1 & x_{11}^m & x_{12}^m & \cdots & x_{1M_1}^m \\ 1 & x_{21}^m & x_{22}^m & \cdots & x_{2M_1}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cdots & x_{ri}^m & \cdots & x_{rM_1}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{M_2 1}^m & x_{M_2 2}^m & \cdots & x_{M_2 M_1}^m \end{bmatrix} \tag{4.1}$$

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 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.

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

$$\sum_{r=1}^{M_2} x_{ri} v_{task}^r > \alpha_i \tag{4.3}$$

The matrix encoding strategy is presented in a simpler form, where the elements of the matrix are either 0 or 1, that is to say, there are only two states, go or not. In the encoding scheme, the driving path of each robot is represented by a row, and a feasible solution is equivalent to a scheduling plan that can efficiently complete all tasks. Furthermore, a binary encoding structure can be conveniently used in evolutionary operations and can efficiently search for potential path combinations.

The decoding procedure is executed as follows: in each row of (4.1), value 1 means the robot visits the task, whereas value 0 represents the robot never visits the task. The order of visiting tasks is determined by

$$VO_{task_i} = \frac{\alpha_i}{s_{path_i}} \tag{4.4}$$

where α_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 α_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 would be encoded as follows:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{4.5}$$

where the first row [1 1 0 1 1 0] means that tasks $[depot, task_1, task_3, task_4]$ needs to be visited sequentially. The values of the relevant parameters of the tasks in Table 1. According to the order $VO_{task_3} > VO_{task_1} > VO_{task_4}$, the visiting path of $robot_1$ is determined as $depot \rightarrow task_3 \rightarrow task_1 \rightarrow task_4$.

Table 1. The values of the relevant parameters of the tasks in Figure 1.

Parameters	$task_1$	$task_2$	$task_3$	$task_4$	$task_5$
α	0.3	0.3	0.5	0.2	0.2
S	3	3	4	8	8
VO_{task}	0.1	0.1	0.125	0.025	0.025

4.1.2. Evolutionary operators

I. Mutation operator

In the proposed follower's algorithm, the total two mutation strategies are adopted in early and later evolution stages, respectively. The first mutation operator is designed as follows:

Mutation operator 1, $DE/rand-to-mean_{better}/1/bin$:

$$V_m = X_{r_1} + rand(0,1)(X_{mean_{better}} - X_{r_1}) + F_{sf}(X_{r_2} - X_{r_3}) \tag{4.6}$$

In the mutation operator, a group of top individuals are chosen to provide an evolutionary direction for individuals. Different from the mutation operator 2 (as below), the mutation operator 1 can keep the diversity of offspring since the heuristic information comes from a group of better individuals, instead of a single point. The advantage of the scheme is that more than one better point can provide a more stable evolutionary direction than that a single point does.

In the later stage of evolution, the population trends to convergence and then only a single best individual can work well. As a result, the following mutation operation 2 is directly adopted.

Mutation operation 2, $DE/rand-to-best/1/bin$:

$$V_m = X_{r_1} + rand(0,1)(X_{best} - X_{r_1}) + F_{sf}(X_{r_2} - X_{r_3}) \tag{4.7}$$

where X_{r_1}, X_{r_2}, and X_{r_3} are randomly selected individuals in the population, $rand(0,1)$ represents a random number that follows an uniform distribution in $[0,1]$, $F_{sf} \in [0,2]$ is the scaling factor, $X_{mean_{best}}$ is the average position of some top individuals, and X_{best} is the best individual found so far.

The offspring of the mutation operator can be described as

$$V_m = \begin{bmatrix} 1 & v_{11}^m & v_{12}^m & \cdots & v_{1M_1}^m \\ 1 & v_{21}^m & v_{22}^m & \cdots & v_{2M_1}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cdots & v_{ri}^m & \cdots & v_{rM_1}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & v_{M_2 1}^m & v_{M_2 2}^m & \cdots & v_{M_2 M_1}^m \end{bmatrix} \tag{4.8}$$

II. Crossover operator

Through the binomial crossover, a trial vector U_m is generated based on X_m and V_m.

$$U_m = \begin{bmatrix} 1 & u_{11}^m & u_{12}^m & \cdots & u_{1M_1}^m \\ 1 & u_{21}^m & u_{22}^m & \cdots & u_{2M_1}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cdots & u_{ri}^m & \cdots & u_{rM_1}^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & u_{M_2 1}^m & u_{M_2 2}^m & \cdots & u_{M_2 M_1}^m \end{bmatrix} \tag{4.9}$$

$$u_{ri}^m = \begin{cases} v_{ri}^m, if\ rand_j \le CR\ or\ j = j_{rand} \\ x_{ri}^m, otherwise \end{cases} \tag{4.10}$$

where $m = 1, 2, \cdots, NP$, $CR \in [0, 1]$ is the crossover control parameter, and $rand_j$ is a random number $[0, 1]$.

III. Individual repair operator

During the evolution of the individual, there may be cases where none of the robots can complete their allocated tasks, leading to an infeasible solution. For example, given an MPDAP instance with two robots and five tasks, the inherent increment rate of a task is larger than the individual ability of both robots. Thus, the robots have to go to the same task and complete it. If only a single robot is asked to execute a task with a large inherent increment rate separately, then such a task will not be completed. To address this issue, a individual repair operator is developed to reallocate the robots to make the solution feasible. The mutation and crossover operators only affect the second to last column of the individual matrix X_m and V_m, so the individual repair operator will only target these columns as well.

In the offspring individual matrix U_m, 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). The repair process is as follows:

Step 1. The elements in the individual matrix are made feasible, i.e., the elements greater than 1 become 1, the elements less than 0 become 0, and others are rounded off.

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 the repair process.

IV. Selection operator

In the process of evolution, two selection operators are used. One is the same as in the original differential evolution, which is denoted by Selection operator 1 and expressed as follows:

$$X_i = \begin{cases} U_i, if\ f_{fit}(U_i) \leq f_{fit}(X_i) \\ X_i, otherwise \end{cases} \tag{4.11}$$

where, f_{fit}, as the fitness, is the follower's objective function .

The other is newly provided as a selection operator 2, which can be finished as follows: in all individuals, parents and offspring, some top individuals are first selected into the next generation of population, and then the rest are chosen by the procedure of selection operator 1. It follows that the selection operator 2 focuses on better individuals instead of diversity kept by parents. Hence the selection operator 2 is helpful to speed up the convergence of the algorithm.

In early evolution, the selection operator 1 is executed for the purpose of maintaining diversity. In order to improve the exploitation capability of the algorithm, the selection operator 2 is utilized in the later stage.

4.1.3. Pseudo-code and flowchart of the follower's algorithm

Based on the mentioned-above design, the pseudo-code of the follower's algorithm is presented in Algorithm 1.

Algorithm 1 Procedure of the proposed follower's algorithm

Require: T^f_{max}: maximum running times; benchmark instance; N^f:population size; Y:leader's variable (act as parameters for follower's problem)

Ensure: output the set A to get the optimal solution X^*

generate initial population $P^f(T^f) = \{X_1, X_2, \cdots, X_{N^f}\}$; $T^f \leftarrow 0$.
store the optimal individual to the set A according to the fitness value of the individuals in $P^f(T^f)$.
while $T^f \leq \alpha T^f_{max}$ **do**
    the offspring population $O^f(T^f)$ is obtained by the mutation operator 1 and crossover operator of population $P^f(T^f)$.
    the next generation population $P^f(T^f + 1)$ is obtained from population $P^f(T^f)$ and offspring population $O^f(T^f)$ according to individual repair operator and the selection operator 1.
    update the set A according to the fitness value of the individuals in $O^f(T^f)$.
    $T^f = T^f + 1$.
end while
while $\alpha T^f_{max} < T^f \leq T^f_{max}$ **do**
    the offspring population $O^f(T^f)$ is obtained by the mutation operator 2 and crossover operator of population $P_f(T^f)$.
    the next generation population $P^f(T^f + 1)$ is obtained from population $P^f(T^f)$ and offspring population $O^f(T^f)$ according to individual repair operator and the selection operator 2.
    update the set A according to the fitness value of the individuals in $O^f(T^f)$.
    $T^f = T^f + 1$.
end while

4.2. A novel differential evolution for the bilevel model (NDE)

In the bilevel optimization algorithm for solving the MPDAP, the leader's objective function is taken as the fitness function, and a classical differential evolution is adopted to optimize the variables of the leader's problem. The pseudo-code of the bilevel optimization algorithm is presented in Algorithm 2.

Algorithm 2 NDE for solving the bilevel optimization model

Require: N^l: population size; T^l_{max}: maximum running times;
Ensure: the best solution Y^{*best}
1: generate initial population $P^l(T^l) = \{Y_1, Y_2, \cdots, Y_{N_u}\}$; $T^l \leftarrow 0$.
2: use Algorithm 1 to solve the corresponding follower's problem for each individual from the population $P^l(T^l)$.
3: store the optimal individual to the set B according to the fitness value of the individuals in $P^l(T^l)$.
4: **while** $T^l \leq T^l_{max}$ **do**
5: the offspring population $O^l(T^l)$ is obtained from $P^l(T^l)$ by the mutation operator and the crossover operator in classical DE.
6: the next generation population $P^l(T^l + 1)$ is obtained from population $P^l(T^l)$ and offspring population $O^l(T^l)$ according to the selection operator in classical DE.
7: use Algorithm 1 to solve the corresponding follower's problem for each individual in population $P^l(T^l + 1)$.
8: update the set B according to the fitness value of the individuals in $P^l(T^l + 1)$.
9: $T^l = T^l + 1$.
10: **end while**

5. Simulation

5.1. Comparison with other methods

All 50 benchmark instances are taken from [11]. In these 50 benchmark instances, the number of robots is fixed, and the task demand function is a linear function. The differences between groups of benchmark instances are shown in Table 2. M_1 is the number of tasks and M_2 is the number of robots. In order to make the experimental results well presented, the benchmark instances were divided into 3 groups, namely Group 1, Group 2, and Group 3. All the instances are called 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, the benchmark instance 1 is named as G_1_5_4_0.39, where 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. In order to make the comparison fair and verify the effectiveness of the NDE, the parameter settings are taken similar to the benchmark instances, and the stopping criterion is set is the same as the competitors. The proposed algorithm is run independently for 30 times, and the computational results are recorded in Tables 3-8. The data of MA [9] based on two variants are shown in columns MA-MLS and MA-OLS in Tables 3, 5, and 7. The data of EDA [10] are shown in column EDA in Tables 3, 5, and 7. The data of ILS [32] are presented in column ILS in Tables 4, 6, and 8. The data of AC-ACO [11] are provided in column AC-ACO in Table 4, 6, and 8. Column NDE provides the computational results by NDE.

Table 2. Differences between groups of benchmark instances.

Instance	The number of instances	Range of M_1	Range of M_2	Range of $M_1 + M_2$
Group 1	27	[4,40]	[3,30]	[0,50)
Group 2	6	[10,60]	[15,40]	[50,95)
Group 3	17	[15,120]	[20,120]	[95,180]

In Tables 3-8, the comparison data include 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 indicate 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 Figure 3 (the Group 1), Figure 4 (the Group 2), and Figure 5 (the Group 3). For the case that some algorithms fail to find the optimal solution on some instances, a maximum value is used on the figure to ensure the visuality of figures.

The results of the Group 1 are shown in Table 3 and Table 4. NDE is slightly worse than MA-MLS in G_1_15_20_0.67, G_1_17_23_1.71, G_1_20_20_0.58, and G_1_20_20_0.967. However, NDE is better than the comparison algorithm in the remaining 23 instances in Table 3. In Table 4, NDE provides worse results for 11 instances, but better results for the remaining 16 instances. Among the 27 benchmark instances in Group 1, the rank-order values of NDE are 31 and 41, respectively. In Table 4, the rank-order values of both AC-ACO and NDE are 41, indicating that they both have similar algorithm performances. Compared with other algorithms except AC-ACO, the rank values of NDE are the smallest.

Table 3. Comparison (I) of experimental results on the Group 1.

Instance	MA-MLS		MA-OLS		EDA		NDE	
	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)	1.3E+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	1.16E+3(1)	3.0E+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)	2.5E+0
G_1_3_15_5.03	1.09E+5(3)	4.4E+4	5.98E+4(2)	1.3E+4	∗(4)	∗	2.75E+4(1)	7.4E+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)	7.5E+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)	1.1E+1
G_1_5_10_3.67	3.12E+4(3)	6.3E+3	2.23E+4(2)	2.9E+3	∗(4)	∗	1.01E+4(1)	2.8E+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)	4.5E+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)	3.2E+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)	3.0E+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)	1.7E+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	3.25E+2(1)	1.1E+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	6.15E+2(1)	4.7E+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)	1.5E+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)	5.0E+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	2.38E+2(1)	3.2E+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)	5.6E+2
G_1_10_20_6.04	∗(2)	∗	∗(2)	∗	∗(2)	∗	8.94E+4(1)	4.23E+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)	2.2E+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)	4.1E+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)	2.3E+0
G_1_15_20_5.98	∗(2)	∗	∗(2)	∗	∗(2)	∗	7.02E+4(1)	1.9E+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)	4E+0
G_1_20_20_0.58	2.74E+2(1)	5.7E+0	2.88E+2(3)	2.5E+0	3.05E+2(4)	4.4E+0	2.86E+2(2)	2.3E+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+29(1)	1.3E+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)	4.5E+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)	2.1E+1
Rank value	60	-	66	-	99	-	31	-

The computational results on the Group 2 are shown in Table 5 and Table 6. In Table 5-6, NDE has slightly worse results for G_2_40_15_0.67, but better results in the remaining 5 instances. For 6 benchmark instances in the Group 2, the rank-order of the NDE are 9 and 8 , respectively. Compared with other algorithms, NDE has a better performance in Group 2.

The computational results on Group 3 are shown in Table 7 and Table 8. In Table 7, the results of NDE are all better than those of the comparison algorithm. Although Table 8 displays that NDE is slightly worse than AC-ACO in G_3_60_30_1.14 and G_3_80_80_0.54, NDE is better than the comparison

Table 4. Comparison (II) of experimental results on the Group 1.

Instance	ILS		AC-ACO		NDE	
	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)	1.3E+0
G_1_5_5_1.66	1.20E+3(2)	5.7E+1	1.23E+3(3)	2.9E+1	1.16E+3(1)	3.0E+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)	5E+0
G_1_3_15_5.03	4.00E+4(3)	7.4E+3	2.73E+4(1)	10.0E+1	2.75E+4(2)	7.4E+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)	7.5E+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)	1.1E+1
G_1_5_10_3.67	1.83E+4(3)	1.8E+3	1.31E+4(2)	7.1E+2	1.01E+4(1)	2.8E+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)	4.5E+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)	3.2E+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)	3.0E+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)	1.7E+0
G_1_15_10_1.17	3.61E+2(3)	2.1E+1	3.42E+2(2)	8.0E+0	3.25E+2(1)	1.1E+0
G_1_10_15_1.3	6.55E+2(3)	3.7E+1	6.30E+2(2)	1.3E+1	6.15E+2(1)	4.7E+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)	1.5E+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)	5.0E+0
G_1_20_10_0.94	2.70E+2(3)	1.6E+1	2.50E+2(2)	3.6E+0	2.38E+2(1)	3.2E+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)	5.6E+2
G_1_10_20_6.04	∗(3)	∗	9.19E+4(2)	4.1E+3	8.94E+4(1)	4.23E+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)	2.2E+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)	4.1E+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)	2.3E+0
G_1_15_20_5.98	∗(3)	∗	7.83E+4(2)	2.1E+3	7.02E+4(1)	1.9E+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)	4E+0
G_1_20_20_0.58	2.73E+2(2)	5.5E+0	2.60E+2(1)	3.7E+0	2.86E+2(3)	2.3E+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)	1.3E+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)	4.5E+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)	2.1E+1
Rank value	73	-	41	-	41	-

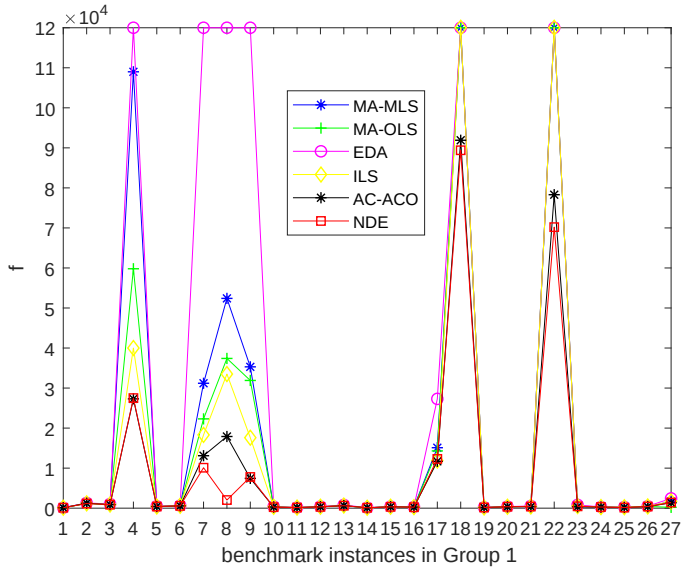

Figure 3. Comparison of experimental results on the Group 1.

Table 5. Comparison (I) of experimental results on the Group 2.

Instance	MA-MLS		MA-OLS		EDA		NDE	
	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)	2.4E+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	3.31E+2(4)	1.5E+0
G_2_20_40_3.61	2.21E+4(2)	3.0E+3	5.28E+4(3)	4.6E+3	∗(4)	∗	4.10E+3(1)	2.2E+1
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	4.24E+2(1)	1.3E+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	8.86E+2(1)	2.5E+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	5.82E+3(1)	3.6E+2
Rank value	11	-	18	-	21	-	9	-

Table 6. Comparison (II) of experimental results on the Group 2.

Instance	ILS		AC-ACO		NDE	
	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(1)	2.4E+0
G_2_40_15_0.67	2.96E+2(2)	1.2E+1	2.80E+2(1)	4.3E+0	3.31E+2(3)	1.5E+0
G_2_20_40_3.61	1.32E+4(3)	4.0E+3	5.53E+3(2)	4.3E+2	4.10E+3(1)	2.2E+1
G_2_30_30_1.04	5.48E+2(3)	4.1E+1	4.66E+2(2)	6.4E+0	4.33E+2(1)	1.3E+1
G_2_30_30_1.94	1.31E+3(3)	7.8E+1	1.10E+3(2)	4.0E+1	8.86E+2(1)	2.5E+1
G_2_15_60_3.64	1.12E+4(3)	1.9E+3	1.01E+4(2)	4.8E+2	5.82E+3(1)	3.6E+2
Rank value	16	-	10	-	8	-

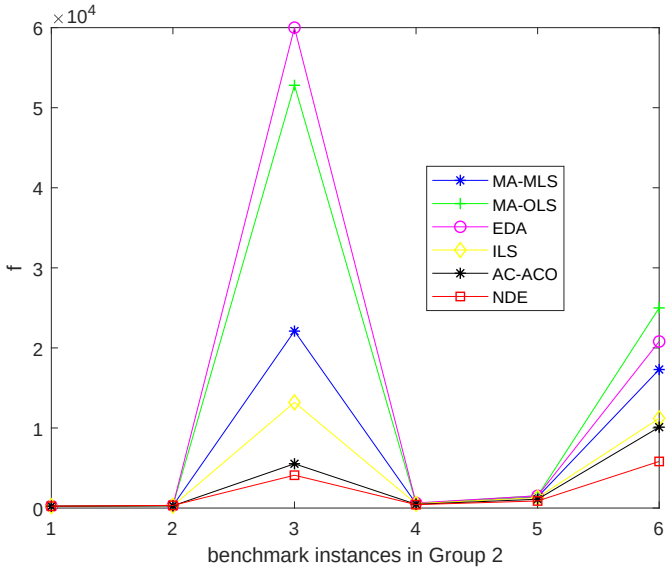

Figure 4. Comparison of experimental results on the Group 2.

algorithm in other 15 instances. For 17 benchmark instances in Group 3, the rank-orders of the NDE are 17 and 19, respectively. Compared with other algorithms, NDE has a better performance in Group 3.

Table 7. Comparison (I) of experimental results on the Group 3.

Instance	MA-MLS		MA-OLS		EDA		NDE	
	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	1.65E+2(1)	2.0E+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	6.76E+2(1)	5.6E+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	2.59E+2(1)	2.9E+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	2.47E+2(1)	3.2E+0
G_3_20_80_4.33	1.25E+5(2)	2.5E+4	2.33E+5(3)	2.0E+4	$*$(4)	$*$	1.90E+4(1)	2.11E+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	5.17E+2(1)	4.6E+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)	2.9E+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)	1.0E+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)	5.2E+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)	4.5E+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)	4.0E+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)	4.0E+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	3.85E+2(1)	7.0E+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	2.65E+2(1)	6.9E+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)	3.1E+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)	2.4E+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)	4.5E+1
Rank value	38	-	56	-	57	-	17	-

Table 8. Comparison (II) of experimental results on the Group 3.

Instance	ILS		AC-ACO		NDE	
	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	1.65E+2(1)	2.0E+0
G_3_20_60_1.51	9.60E+2(3)	2.7E+1	7.53E+2(2)	1.9E+1	6.76E+2(1)	5.6E+0
G_3_80_20_0.7	3.31E+2(3)	1.5E+1	2.86E+2(2)	5.5E+0	2.59E+2(1)	2.9E+0
G_3_80_20_1.12	4.16E+2(3)	2.2E+1	3.10E+2(2)	7.2E+0	2.47E+2(1)	3.2E+0
G_3_20_80_4.33	6.42E+4(3)	5.2E+3	2.21E+4(2)	2.0E+3	1.90E+4(1)	2.11E+2
G_3_60_30_1.14	6.21E+2(3)	3.7E+1	5.08E+2(1)	7.2E+0	5.17E+2(2)	4.6E+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)	2.9E+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)	1.0E+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)	5.2E+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)	4.5E+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)	4.0E+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)	4.0E+0
G_3_60_60_0.922	5.58E+2(3)	1.5E+1	4.18E+2(2)	6.9E+0	3.85E+2(1)	7.0E+0
G_3_120_30_1.2	4.97E+2(3)	1.8E+1	3.40E+2(2)	6.7E+0	2.65E+2(1)	6.9E+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)	3.1E+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)	2.4E+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)	4.5E+1
Rank value	51	-	32	-	19	-

NDE and AC-ACO have similar performance in solving benchmark instances in the Group 1; however NDE has better performance in solving benchmark instances in the Group 2 and the Group 3. Therefore, it is not difficult to see that the NDE is effective for solving these 50 benchmark instances.

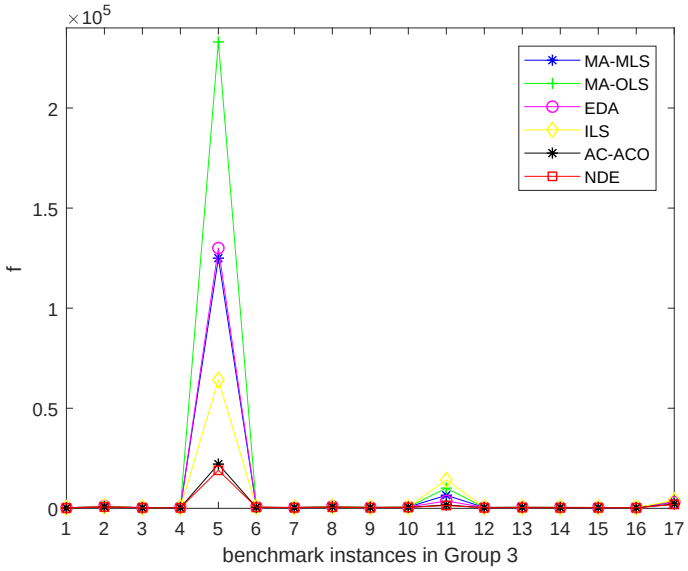

Figure 5. Comparison of experimental results on the Group 3.

5.2. Experimental results under the bilevel optimization model

In this part, when the bilevel optimization model is taken into account, it is very necessary to analyse the effect on the total cost caused by the amount of robots. For the 50 benchmark instances in subsection 5.1., besides the same amount of robots as in literature, some additional robots are added to perform these tasks. The purpose of the procedure is to understand the optimal amount of robots, such that the total cost can be minimized. When robots need to be added, the robot with the lowest efficiency is first chosen since it can lead to the worse case, increase the cost, and lower the efficiency. The experimental results are shown in Tables 9 to 11. In Tables 9 to 11, F represents the leader's objective function in the bilevel programming model, f represents the follower's objective function, and N_i represents the number of robots added to execute tasks.

Since there are multiple robots with different capabilities in the benchmark instances, in line with the working capability of robots, the state-determined cost of renting (or buying) robots is set as the value that equals to λ_2 times the working efficiency of the robots, and the running cost of the robot is the product of working efficiency and working time. Without any loss of generality, the values of both λ_1 and λ_2 are taken as 1 in the experiment.

Among the 27 benchmark instances in Group 1 in Table 9, based on the bilevel optimization model, shows that the optimal solution of 9 instances is better than that of the original instances. For example, for G_1_5_5_1.66, the optimal solution needs to add 3 robots to the original instances. When the number of robots is increased, the state-determined cost of robots increases, but the time to complete the task can evidently decrease. As a result, the leader's objective function is decreased. So the decision maker can choose the better option.

For the 6 instances in Group 2 in Table 10, we find that on two instances (G_2_20_40_3.61 and G_2_15_60_3.64), the better solutions are obtained when an appropriate amount of robots are added. For the benchmark instances in Group 3, from Table 11, one can find that one instance G_2_20_80_4.33 finds

Table 9. The solutions of bilevel programming model for Group 1.

Instance	best solution of bilevel model			best solution of original instance		
	F	f	N_i	F	f	N_i
G_1_5_4_0.39	7.39E+2	1.03E+2	0	7.39E+2	1.03E+2	0
G_1_5_5_1.66	9.54E+2	7.64E+2	3	1.01E+3	1.16E+3	0
G_1_3_10_1.51	4.46E+2	5.50E+2	1	4.77E+2	8.76E+2	0
G_1_3_15_5.03	4.78E+3	1.71E+3	3	1.10E+3	2.75E+4	0
G_1_5_10_0.93	3.34E+3	4.09E+2	0	3.34E+3	4.09E+2	0
G_1_10_5_1.39	4.07E+3	5.65E+2	0	4.07E+3	5.65E+2	0
G_1_5_10_3.67	3.79E+3	2.84E+3	3	5.34E+3	1.01E+4	0
G_1_5_20_4.36	5.62E+3	1.18E+3	1	6.18E+3	2.03E+2	0
G_1_10_10_3.79	7.28E+3	2.56E+3	6	9.37E+3	7.79E+3	0
G_1_11_11_1.28	1.88E+4	2.50E+2	0	1.88E+4	2.50E+2	0
G_1_30_5_0.46	1.19E+4	1.50E+2	0	1.19E+4	1.50E+2	0
G_1_15_10_1.17	5.74E+3	3.25E+2	0	5.74E+3	3.25E+2	0
G_1_10_15_1.3	3.97E+3	6.15E+2	0	3.97E+3	6.15E+2	0
G_1_20_10_0.47	1.32E+4	9.77E+1	0	1.32E+4	9.77E+1	0
G_1_20_10_0.96	7.25E+3	3.31E+2	0	7.25E+3	3.31E+2	0
G_1_20_10_0.94	7.25E+3	2.38E+2	0	7.25E+3	2.38E+2	0
G_1_5_40_3.95	8.61E+3	4.96E+3	2	1.16E+4	1.23E+4	0
G_1_10_20_6.04	3.28E+4	3.81E+4	2	6.37E+4	8.94E+4	0
G_1_30_10_0.65	1.09E+4	1.68E+2	0	1.09E+4	1.68E+2	0
G_1_15_20_0.67	1.00E+4	3.60E+2	0	1.00E+4	3.60E+2	0
G_1_30_10_1.34	1.10E+4	4.02E+2	0	1.10E+4	4.02E+2	0
G_1_15_20_5.98	2.70E+4	1.78E+4	5	7.23E+4	7.02E+4	0
G_1_17_23_1.71	1.61E+5	4.58E+2	0	1.61E+5	4.58E+2	0
G_1_20_20_0.58	1.29E+4	2.86E+2	0	1.29E+4	2.86E+2	0
G_1_20_20_0.97	2.39E+5	1.45E+2	0	2.39E+5	1.45E+2	0
G_1_20_20_0.967	1.39E+4	4.08E+2	0	1.39E+4	4.08E+2	0
G_1_15_30_2.16	6.33E+3	1.41E+3	0	6.33E+3	1.41E+3	0

Table 10. The solutions of bilevel programming model for Group 2.

Instance	best solution of bilevel model			best solution of original instance		
	F	f	N_i	F	f	N_i
G_2_40_10_0.67	1.41E+4	2.29E+2	0	1.41E+4	2.29E+2	0
G_2_40_15_0.67	1.49E+4	3.31E+2	0	1.49E+4	3.31E+2	0
G_2_20_40_3.61	1.27E+4	2.97E+3	2	1.42E+4	4.10E+3	0
G_2_30_30_1.04	1.15E+4	4.33E+2	0	1.15E+4	4.33E+2	0
G_2_30_30_1.94	1.19E+4	8.86E+2	0	1.19E+4	8.86E+2	0
G_2_15_60_3.64	1.12E+4	4.38E+3	2	1.19E+4	5.82E+3	0

Table 11. The solutions of bilevel programming model for Group 3.

Instance	best solution of bilevel model			best solution of original instance		
	F	f	N_i	F	f	N_i
G_3_80_15_0.76	2.89E+4	1.65E+2	0	2.89E+4	1.65E+2	0
G_3_20_60_1.51	1.49E+4	6.76E+2	0	1.49E+4	6.76E+2	0
G_3_80_20_0.7	2.85E+4	2.59E+2	0	2.85E+4	2.59E+2	0
G_3_80_20_1.12	2.81E+4	2.47E+2	0	2.81E+4	2.47E+2	0
G_3_20_80_4.33	2.74E+4	1.37E+4	2	4.46E+3	1.90E+4	0
G_3_60_30_1.14	3.35E+4	5.17E+2	0	3.35E+4	5.17E+2	0
G_3_60_40_0.69	2.14E+4	3.20E+2	0	2.14E+4	3.20E+2	0
G_3_40_60_1.54	2.79E+4	5.37E+2	0	2.79E+4	5.37E+2	0
G_3_80_40_0.97	2.83E+4	3.10E+2	0	2.83E+4	3.10E+2	0
G_3_40_80_1.05	2.78E+4	4.60E+2	0	2.78E+4	4.60E+2	0
G_3_80_40_2.25	3.45E+4	1.42E+3	0	3.45E+4	1.42E+3	0
G_3_60_60_0.92	2.38E+4	2.79E+2	0	2.38E+4	2.79E+2	0
G_3_60_60_0.922	2.21E+4	3.85E+2	0	2.21E+4	3.85E+2	0
G_3_120_30_1.2	4.11E+4	2.65E+2	0	4.11E+4	2.65E+2	0
G_3_80_60_0.72	3.01E+4	3.17E+2	0	3.01E+4	3.17E+2	0
G_3_80_80_0.54	5.42E+4	2.50E+2	0	5.42E+4	2.50E+2	0

the better solution than original case.

6. Discussion

In the field of task scheduling and route optimization, the multi-point dynamic aggregation problem is interesting but computationally complex. In the present research, in order to reduce decision-making cost, some key parameters are predetermined, which should be further optimized, such as the number of robots as well as tasks. Based on these assumptions, some simplified models have be given, which simply focus on optimizing the routes of robots to minimize the completion time of all tasks. In fact, from the decision maker's point of view, at least two additional costs, robots and property damage, need to be taken into account. It follows that optimization procedure should be divided into two levels. One is to determine the number of robots(leader's level), whereas the other is to optimize the routes of these robots(follower's level). As a result, a bilvel programming problem is proposed in the manuscript. When the leader's variables are given, the problem can degrade into a single-level problem just as in literature. Hence, the proposed bilevel problem is more universal than the existing models.

Bilevel programming problems are computationally hard. In order to efficiently deal with the hierarchical optimization problem, a two-level DE is developed. In the algorithm for the follower's program, a matrix encoding scheme is provided and followed by a newly designed decoding rule. The individual representation is in nature binary, which is simpler and more efficient than ever. In addition, the mean location of multiple high-qualify individuals can provide a stabler evolutionary direction than a single elite individual.

In spite of the fact that the bilevel model is more suitable for practice, because of the discrete variables and hierarchical structure involved, this problem is still very difficult to solve. More specifically, for large-scale cases in which too many robots and tasks are involved, the proposed algorithm still needs to pay expensive computation cost.

7. Conclusion

In the real-world field, MPDAP is a challenging and difficult optimization problem. In this manuscript, the problem is re-modeled as a bilevel programming model by considering the total cost of completion tasks and the damages. In order to efficiently deal with the bilevel programming problem, a novel differential evolution with a nested structure is developed, in which a new matrix encoding method is designed to represent the routes of robots. Meanwhile, in order to solve the follower's problem efficiently, we also present two-strategy mutation and selection operators. From the experimental results, one can find that the proposed algorithm has better performance when compared with other similar algorithms. Also, based on the bilevel optimization model, the solutions with smaller cost are obtained by dispatching additional robots.

When some interesting topics, such as the dynamic task number and the priority of tasks, are taken into the model, the optimization procedure may become more complex than ever. In the future study, more efficient heuristical operators as well as optimization techniques need to be further developed for this kind of problems.

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 conflict 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 *Proceedings IEEE International Conference on Control and Automation*, Kathmandu, Nepal, 2016, 933–938. https://doi.org/10.1109/ICCA.2016.7505398
2. V. Akbari, F. S. Salman, Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity, *European J. Operat. Res.*, **257** (2017), 625–640. https://doi.org/10.1016/j.ejor.2016.07.043
3. A. Khan, B. Rinner, A. Cavallaro, Cooperative robots to observe moving targets: Review, *IEEE Transact. Cybernet.*, **48** (2018), 187–198. https://doi.org/10.1109/TCYB.2016.2628161
4. 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

5. R. P. Yuan , J. T. Li , X. L. Wang , L. Y. He, Multirobot task allocation in e-commerce robotic mobile fulfillment systems, *Math. Problems Eng.*, (2021), Article ID 6308950. https://doi.org/10.1155/2021/6308950
6. C. Y. Ju, J. Kim , J. Seol , H. I. Son, A review on multirobot systems in agriculture, *Comput. Electron. Agriculture*, **202** (2022), 107336. https://doi.org/10.1016/j.compag.2022.107336
7. B. Xin, S. Liu, Z. Peng, G. Gao, An estimation of distribution algorithm for multi-robot multi-point dynamic aggregation problem, in *Proceedings IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Miyazaki, Japan, 2018, 775–780.
8. 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.
9. G. 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 aggre- gation missions, in*Proceedings IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, U.K., 2020, 1–8.
10. 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 Genetic Evolution Computational Conference Companion*, 2018, 251–252.
11. G. Q. Gao, Y. Mei, Y. H. Jia, W. N. Browne, B. Xin, Adaptive coordination ant colony optimization for multipoint dynamic aggregation, *IEEE Transact. Cybernet.*, **52** (2022), 7362–7376. https://doi.org/10.1109/TCYB.2020.3042511
12. A. Sinha, P. Malo, K. Deb, A review on bilevel optimization: From classical to evolutionary approaches and applications, *IEEE Transact. Evolut. Comput.*, **22** (2018). https://doi.org/10.1109/TEVC.2017.2712906
13. A. Sinha, T. Soun, K. Deb, Using Karush-Kuhn-Tucker proximity measure for solving bilevel optimization problems, *Swarm and Evolutionary Computation*, **44** (2019), 496-510. https://doi.org/10.1016/j.swevo.2018.06.004
14. S. N. Liu, M. Z. Wang, N. Kong, An enhanced branch-and-bound algorithm for bilevel integer linear programming, *European J. Operat. Res.*, **291** (2020), 61–67. https://doi.org/10.1016/j.ejor.2020.10.002
15. G. X. Li, X. M. Yang, Convexification method for bilevel programs with a nonconvex follower's problem, *J. Optimiz. Theory Appl.*, **188** (2021), 724–743. https://doi.org/10.1007/s10957-020-01804-9
16. A. Joseph, O. Y. Ozaltin, Feature selection for classification models via bilevel optimization, *Comput. Operat. Res.*, **5** (2018), 1–32. https://doi.org/10.1016/j.cor.2018.05.005
17. H. C. Li, Z. C. Wang, An evolutionary algorithm using parameter space searching for interval linear fractional bilevel programming problems,*Int. J. Pattern Recogn. Artif. Intell.*, **30** (2016), 1–18. https://doi.org/10.1142/S0218001416590114
18. Y. Aboelnage, S. Nasr, Modified evolutionary algorithm and chaotic search for bilevel programming problems, *Symmetry*, **12** (2020), 767–796. https://doi.org/10.3390/sym12050767

19. N. N. Goshu, S. M. Kassa, A systematic sampling evolutionary (SSE) method for stochastic bilevel programming problems, *Comput. Operat. Res.*, **120** (2020), 1–14. https://doi.org/10.1016/j.cor.2020.104942

20. M. M. Islam, H. K. Singh, T. Ray, An enhanced memetic algorithm for single-objective bilevel optimization problems, *Evolut. Comput.*, **25** (2017), 607–642. https://doi.org/10.1162/EVCOa00198

21. A. Yousria, S. Nasr, I. El-Desoky, Enhanced genetic algorithm and chaos search for bilevel programming problems, *In Advances in Intelligent Systems and Computing*, Hassanien, A.Ed. Springer: Cham, Switzerland, 2019, 478–487.

22. A. Sinha, Z. Lu, K. Deb, P. Malo, Bilevel optimization based on iterative approximation of multiple mappings. *Journal of Heuristics*, **26** (2020), 151–185. https://doi.org/10.48550/arXiv.1702.03394

23. A. Molai, Solving fuzzy multiobjective linear bilevel programming problems based on the extension principle, *Iranian J. Numer. Anal. Optimiz.*, **11** (2021), 1–31. https://doi.org/10.22067/IJNAO.2021.11304.0

24. R. Storn, K. Price, Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimiz.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328

25. S. Chakraborty, A. Kumar, S. Absalom, E. Ezugwu, Differential evolution and its applications in image processing problems: a comprehensive review, *Arch. Comput. Methods Eng.*, **30** (2023), 985-–1040. https://doi.org/10.1007/s11831-022-09825-5

26. Y. Wang, H. Liu, H. Long, Z. Zhang, S. Yang, Differential evolution with a new encoding mechanism for optimizing wind farm layout, *IEEE Transact. Industr. Inform.*, **14** (2018), 1040–1054. https://doi.org/10.1109/TII.2017.2743761

27. R. Chi, Z. Li, X. Chi, Z. Qu, H. Tu, Reactive power optimization of power system based on improved differential evolution algorithm, *Math. Problems Eng.*, (2021), 1–19. https://doi.org/10.1155/2021/6690924

28. L. Jebaraj, C. Venkatesan, I. Soubache, C. C. A. Rajan, Application of differential evolution algorithm in static and dynamic economic or emission dispatch problem: A review, *Renewable Sustain. Energy Rev.*, **77** (2017), 1206–1220. https://doi.org/10.1016/j.rser.2017.03.097

29. X. Sui, S. C. Chu, J. Pan, H. Luo, Parallel compact differential evolution for optimization applied to image segmentation, *Appl. Sci.*, **10** (2020), 2195. https://doi.org/10.3390/app10062195

30. M. Abdel-Basset, R. Mohamed, W. Elkhalik, M. Sharawi, Task scheduling approach in cloud computing environment using hybrid differential evolution, *Mathematics*, **10** (2022), 4049. https://doi.org/10.3390/math10214049

31. S. Tsafarakis, K. Zervoudakis, A. Andronikidis, E. Altsitsiadis, Fuzzy self-tuning differential evolution for optimal product line design, *European J. Operat. Res.*, **287** (2020), 1161–1169. https://doi.org/10.1016/j.ejor.2020.05.018

32. X. Wang, T. M. Choi, Z. Li, S. Shao, An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem, *IEEE Transact. Syst. Man Cybern. Syst.*, **50** (2020), 4948–4958. https://doi.org/10.1109/TSMC.2019.2938298

AIMS Press

© 2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)