Multi-Agent Task Allocation Based on Discrete DEPSO in Epidemic Scenarios

XINYAO MA1,2, CHUNMEI ZHANG1,2, FENGLIN YAO3, AND ZHANLONG LI3

1School of Electronic Information, Taiyuan University of Science and Technology, Taiyuan, Shanxi 030024, China
2Shanxi Key Laboratory of Advanced Control and Equipment Intelligence, Taiyuan, Shanxi 030024, China
3School of Mechanical Engineering, Taiyuan University of Science and Technology, Taiyuan, Shanxi 030024, China

Corresponding authors: Chunmei Zhang (2005085@tyust.edu.cn) and Fenglin Yao (2005084@tyust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 52272401 and Grant 61603266, and in part by the Natural Science Foundation of Shanxi Province under Grant 202203021211198.

ABSTRACT Multi-Agent Task Allocation is an emerging technology that changes the world in the epidemic scenario through its power to serve the needs of any hospital that requires unmanned operation. In this environment, the end user may want to have a better quality of unmanned service at low loss and high efficiency. We defined a new multi-agent task allocation problem (MATAP) in the epidemic scenario, and then MATAP was formulated. This paper presents a novel hybrid discrete approach that is based on the Differential Evolution Algorithm (DE) and Partial Swarm Optimization (PSO), namely D-DEPSO, for handling this problem. First, the initial personal population was handled by “mutation operation”. Modulus operations in the “mutation operation” modify the numerical overflow of a variable. Second, when updating the speed matrix, the speed matrix is discretized using the “round” function we have defined. Then, a random permutation was used to delete repeated numbers and to reinsert integers in the “crossover operation”. The diversity of the population was expanded by introducing the discrete mutation operation of the DE into the PSO and preserving the optimal solution for each generation using the properties of PSO. It can be used for optimizing a single objective function. Experimental results are compared with other existing metaheuristic algorithms, such as discrete DE, discrete PSO, improved discrete DE, improved discrete PSO, and improved discrete genetic algorithm, in terms of running time and loss. The experiments show that the optimal solutions obtained by D-DEPSO are better than those obtained by other five algorithms. For the actual problem, D-DEPSO can generate an optimal solution by optimal parameter setting to allocate tasks rationally. It can achieve a rational distribution of tasks in the prevention of disease.

INDEX TERMS Multi-agent, task allocation problem, metaheuristic algorithms, D-DEPSO, epidemic scenario.

I. INTRODUCTION

In the last few years, because of the influence of COVID-19, reducing contact between people and people in the epidemic environment is particularly important. Meanwhile, internet of things and artificial intelligence technologies have made great progress in the past decade, benefit from this, multi-agent systems (MAS) [1] begin to be widely applied in reality, such as smart city [2], smart manufacturing [3], unmanned systems [4], smart transportation systems [5], unmanned aerial vehicle (UAV) format combat systems [6]. Hence, major medical institutions have begun to use intelligent robots to gradually replace simple artificial operations [7]. In this specific scenario, agents are regularly allocated different tasks [8], such as medicine distribution, medical material handling, periodic disinfection, long-distance measurement of body temperature, patient supervision, and so on. As COVID-19 spreads globally, the needs of medical institutions around the world are showing a significant growth in the demand for medical robots. Task allocation is an important part of the multi-agent system. Hence, the multi-agent task allocation problem is especially critical.
Due to the sudden emergence of COVID-19, medical resources are at a sharp shortage, so governments must establish temporary hospitals. Because of the environmental complexity, abnormal communication signals, lack of medical staff and task diversity of temporary hospitals, the best way for temporary hospitals is the application of MAS [7]. As reported [9], robots are wirelessly connected to the network at all times, so they can listen to the server from anywhere in the building and accept more than one task along their way, but temporary hospitals need higher reliability. They have to use pre-programmed robots so that one robot executes one task at a time to avoid communication failures in the tough environment.

The Task Allocation Problem (TAP) [10] originally addressed the task of distributing programs among different processors of a distributed computer system to reduce program turnaround time and increase system throughput. And then, with the development of multi-agent systems [1], a single agent has been unable to fulfill the military and civilian fields, and the gradual development of multi-agent and group agent cooperation systems makes the task allocation gradually applicable to the field. It has been demonstrated that finding an optimal solution is an NP-hard problem [11], [12], [13]. There can’t be a good exact algorithm for the optimal solution in polynomial time. The goal of task allocation is to optimize the performance of multi-agent or loss of task execution, such as increasing the number of successfully executed tasks and decreasing task execution time and resource consumption. Furthermore, to optimize the objective model, a vast number of constraint conditions must be satisfied during task allocation. For example, because of their differences in structure, function, and performance, different types of agents are utilized to play different roles and handle diverse tasks. An agent dedicated to distributing medicine can only perform a medicine distribution task, a patrol disinfection agent can only perform a periodic disinfection task, a material handling agent can only perform a medical material handling task, and so on.

In the last few years, many researchers have devoted themselves to the study of MATAP, and many methods have been proposed. Task allocation strategies are divided into two types: exact methods and heuristic algorithms. For small-scale task allocation, exact methods [14] often provide an effective solution. When the scale of the task is extended, exact methods usually fail. In contrast to finding the exact solution, meta-heuristic algorithms typically find sub-optimal solutions. Heuristic algorithms have been researched more thoroughly than exact methods and have been shown to provide an efficient foundation for achieving sub-optimal solutions. Zhang Chunmei presents a distributed memetic differential evolution algorithm to solve the discrete problem [15]. Xin Bin presents a review about the way of hybridizing differential evolution and partial swarm optimization [16]. There have been several achievements in the field of meta-heuristic algorithms for the multi-agent task allocation problem. J Schwarzrock presents swarm intelligence [17] to solve task allocation problem in multi unmanned aerial vehicles systems, its principle is similar to MATAP. Jing Zhou presents a distributed many-objective evolutionary algorithm with greedy algorithm (GA) [18] to solve multi-agent task allocation problems, but the mathematical model for the epidemic scenario has not been fully considered in this paper. So we proposed a new mathematical model to address the epidemic scenario. For the meta-heuristic algorithm, it is easy to fall into a local optimal solution, thus missing the global optimal solution. Wang Lu presents a novel task allocation method named Collection Path Ant Colony Optimization (CPACO) [19] to solve the problem that is easy to converge to the local optimal solution, but it depends on the set of initial parameters selected. There is instability in the results of the algorithm. Dai Jing presents a differential evolution algorithm (DE) [20] to handle the allocation problem of multi-heterogeneous UAV cooperation, but in this scenario, the number of tasks considered is too small to satisfy the requirements of multi-tasks in the epidemic scenario. Maroua Noura proposed a distributed partial swarm optimization (DSPSO) [21] is used for flexible job-shop scheduling problems. It demonstrates the effectiveness of DSPSO in handling discrete problems. Feng Zhang improved quantum particle swarm optimization (QPSO) [22] to handle the task allocation in MAS. It enhances the diversity of the population and makes it have stronger search abilities, but it does not consider the application of QPSO in complex task scenarios. The above are all the static problems, Junier Caminha Amorim [23] presents a swarm-GAP based solution for the task allocation problem in dynamic scenarios.

In view of the above, MATAP in epidemic scenario is a discrete problem. At the same time, it is also a combinatorial optimization problem. A simple discretization of the commonly used continuity algorithm is not applicable to this problem. And there is no mathematical model for MATAP in the epidemic scenario and there is no algorithm for a large discretization problem in the epidemic scenario. The purpose of this paper is to build a mathematical model and to design a discrete hybridizing algorithm for the epidemic scenario to maximize the number of successfully executed tasks, minimize the loss of performing tasks, and minimize task execution time and resource consumption. Overall, the main contributions of our study are listed as follows:

1) In the epidemic scenario, we introduced and defined the mathematical model of MATAP. The model’s objective function minimized the loss of allocation. And we minimized the loss of allocation by selecting appropriate potential agents to execute corresponding tasks and then determining the order of execution.

2) A new approach was presented by combining the discrete DE and discrete PSO algorithms. We apply this approach to solve the task allocation problem in multi-agent systems. According to the urgency of the epidemic scenario and the requirement for minimal allocation losses, the algorithm mainly emphasized rapidity and minimization of results. Therefore, we innovatively combined discrete DE
and discrete partial swarm optimization by designing an algorithm framework, crossover operation, and a method of updating speed and position variables.

(3) Multiple simulation experiments are implemented to demonstrate the effectiveness of the proposed algorithm. The results showed that our algorithm is more effective at solving MATAP in the epidemic scenario.

The remainder of this paper is arranged as follows: In Section II, the MATAP is defined, and its mathematical model is introduced. In Section III, an illustrative example of MATAP is presented to describe how it works. In Section IV, the frameworks of the DE and the PSO are briefly presented, and the procedure of the D-DEPSO is sufficiently described. Section V shows a lot of experimental data and analysis. In Section VI, some conclusions and future prospects of the research are discussed.

II. MATHEMATICAL MODEL FOR MULTI-AGENT TASK ALLOCATION PROBLEM

In order to better explain the task allocation problem of MAS in the epidemic scenario, it needs to be formally described. To build a mathematical model, we refer to the paper [24]. In this paper, all parameters are known and ignored timing by default in MATAP, and two metrics were prioritized: the probability that a medical agent completes a task on time and the quality of task allocation at the same time. Create a system with $N$ medical robots and $T$ different tasks, assuming a scenario with five different tasks to be allocated by five robots. Tasks include medicine distribution, medical material handling, periodic disinfection, long distance measurement of body temperature, and patient supervision. So, $N = 5$, an initial individual of the integer arrangement created by D-DEPSO: $x = [3 \ 1 \ 2 \ 5 \ 4]$, where the number of integer position represents the number of the robot and the integer represents the number of the task. In this example, 3 represents task 3 allocated to robot 1, 1 represents task 1 allocated to robot 2, and so on. As the initial individual of the D-DEPSO algorithm, create random permutations of integers $1 \sim N$, $x_i (i = 1, 2, \ldots, NP$, where $NP$ is the population size).

$$
\begin{align*}
\text{min} \ f(Y) & = \sum_{k=1}^{K} v_k \left\{ \prod_{i=1}^{A} \prod_{j=1}^{T} (1 - p_{ij}) y_{ij} \right\} \\
\text{s.t.:} & \sum_{i=1}^{A} y_{ij(t)} = 1, \quad (i = 1, 2, \ldots, A) \\
& \sum_{j=1}^{T} y_{ij(t)} = 1, \quad (i = 1, 2, \ldots, T) \\
\text{subject to:} & A = T = N
\end{align*}
$$

$p_{ij}$ represents the probability of robot $i$ completing task $j$ on time, $y_{ij}$ represents whether robot $i$ executes task $j$. If $y_{ij} = 1$, it means robot $i$ executes task $j$, or not. So $(1 - p_{ij})y_{ij}$ represents the probability that robot $i$ does not execute task $j$ on time.

The objective Function (1) is to minimize the loss of MATAP and ensure the quality of task allocation at the same time. Create a system with $i$ Agents $A = 1, 2, \ldots, i$, $j$ Tasks $T = 1, 2, \ldots, j$ and the value of Tasks $v_k = 1, 2, \ldots, K$. Because $p_{ij}$ represents probability of robot $i$ finishes task $j$ on time, $1 - p_{ij}$ represents probability of robot $i$ not completes task $j$ on time. $y_{ij} = 1$ represents if robot $i$ is allocated task $j$ to execute; otherwise $y_{ij} = 0$. $v_k$ represents the value of task $j$ completed by robot $i$.

Constraint condition (2) guarantees every task is only executed by one robot. The constraint condition (3) guarantees every robot can only implement one task. Constraint condition (4) guarantees each task is specific to a particular robot only, and each robot is only assigned one task. In the next section, we describe how an initial solution generated by D-DEPSO maps to a 0-1 matrix through an explicit example.

III. ILLUSTRATIVE EXAMPLE OF MATAP

In this section, the working principle of MATAP was interpreted specifically through an illustrative example. Let’s assume a scenario with five different tasks to be allocated by five robots. Tasks include medicine distribution, medical material handling, periodic disinfection, long distance measurement of body temperature, and patient supervision. So, $N = 5$, an initial individual of the integer arrangement created by D-DEPSO: $x = [3 \ 1 \ 2 \ 5 \ 4]$, where the number of integer position represents the number of the robot and the integer represents the number of the task. In this example, 3 represents task 3 allocated to robot 1, 1 represents task 1 allocated to robot 2, and so on. As the initial individual of the D-DEPSO algorithm, create random permutations of integers $1 \sim N$, $x_i (i = 1, 2, \ldots, NP$, where $NP$ is the population size).

$$
Y = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}
$$
Although robots are universal and a machine can perform different tasks, e.g. medicine distribution, medical material handling, or periodic disinfection, the economic loss incurred when performing different tasks is different because the \( v_k \) (i.e., value) of the task is different. Different tasks performed by the same agent result in different economic losses. The \( v_k \) of the task is different. Different tasks performed by the same agent result in different economic losses.

The first row represents medicine distribution; the second row represents periodic disinfection; the third row represents long-distance measurement of body temperature; the fifth row represents patient supervision. The matrix (5) was transferred by the initial individual solution: \( x = [3 1 2 5 4] \). Due to the emergency nature of COVID-19, every robot can only complete one task at a time in this scenario. Using integer arrangements to create multi-agent task pairs as the initial individual generated by D-DEPSO. In matrix (5), the number of rows where “1” is located denotes the number of robots; the number of columns where “1” is located represents the number of tasks.

**IV. D-DEPSO**

In this paper, a discrete differential evolution algorithm and a discrete partial swarm optimization were mainly combined to solve the above multi-agent task optimization in the epidemic scenario. Differential evolution mutation operations are used for mutating personal optimal position, and partial swarm optimization is mainly used to record personal optimal positions and update speed values. The pseudo-code of the D-DEPSO algorithm is shown in Algorithm 1, and parameters related to the D-DEPSO algorithm are shown in Table 2.

**Algorithm 1 D-DEPSO**

**Input:** coefficient of inertia weight \( (w) \), acceleration constant \( (c_1) \), acceleration constant \( (c_2) \), mutation operator \( (f) \), population number \( (NP) \), maximum number of iterations \( (T) \).

**Output:** \( gbest \)

1. randomly initialize matrix \( x \) of position value
2. randomly initialize matrix \( v \) of speed value
3. initialize personal best position \( pbest \) (see section IV-B)
4. initialize global best position \( gbest \) (see section IV-B)
5. record \( gb \)
6. for \( i = 1 : T \) do
7. \( \text{for } j = 1 : NP \) do
8. update matrix of \( p \) and \( pbest \)
9. \( z = x(j,:) \)
10. convert \( z \) to a 0-1 matrix of \( y \)
11. if \( func(y) < pbest[j] \) then
12. \( p(j,:) \leftarrow x(j,:) \)
13. \( pbest(j) \leftarrow func(y) \)
14. end if
15. update \( g \) and \( gbest \)
16. if \( pbest(j) < gbest \) then
17. \( g \leftarrow p(j,:) \)
18. \( gbest \leftarrow pbest(j) \)
19. end if
20. adapted operator (see Section IV-C)
21. \( r1 \leftarrow randi(NP) \)
22. while \( r1 == j \) do
23. \( r1 \leftarrow randi(NP) \)
24. end while
25. \( r2 \leftarrow randi(NP) \)
26. while \( r2 == j \) || \( (r1 == r2) \) do
27. \( r2 \leftarrow randi(NP) \)
28. end while
29. \( r3 \leftarrow randi(NP) \)
30. while \( r3 == j \) || \( (r2 == r3) || (r1 == r3) \) do
31. \( r3 \leftarrow randi(NP) \)
32. end while
33. mutation operation on \( p(j,:) \) (see Section IV-C)
34. crossover operation on \( p(j,:) \) (see Section IV-D)
35. calculate speed variable \( v(j,:) \) (see Section IV-E)
36. for \( m = 1 : NP \) do
37. if \( v(j,m) > NP \) then
38. \( v(j,m) \leftarrow 0 \)
39. else
40. \( v(j,m) \leftarrow v(j,m) \)
41. end if
42. end for
43. crossover operation on \( v(j,:) \)
44. calculate position variable \( x(j,:) \) (see Section IV-E)
45. \( gb(i) \leftarrow gbest \)
46. end for
47. end for
48. end for

131184
A comparison of this point is a personal best value. It will be saved in \( p_{best} \) by using the formula that is shown in Section II. The result at the method present in Section III, and calculate the result for \( y \) the value of the \( p \) matrix to the \( g_{best} \). The flow chart of D-DEPSO algorithm.

**C. MUTATION OPERATION**

In the DE algorithm [25], initialization, mutation strategies, and crossover operations have a significant effect on the diversity of population solutions, thus affecting the quality of solutions. Considering mutation operation and crossover operation as the most important parts in designing our D-DEPSO algorithm. Meanwhile, in order to improve the population diversity in the early stages of the algorithm and to preserve the good solutions in the later stages to avoid the destruction of the optimal solutions, we have added the adapted operator. Its formula is as follows:

\[
\text{lambda} = \exp(1 - T/(T + 1 - i))
\]

\[
F = f * 2^{\text{lambda}}
\]

The adapted operator is depicted in equation (6), where \( T \) represents the maximum number of iterations, \( i \) represents the current number of iterations, and \( f \) represents the initial mutation rate. At first, \( i = 1, F = 2f \), the algorithm can maintain the individual diversity. As the number of loop iterations increases, \( i \) gradually converges to \( T \), and \( F \) gradually converges to \( f \). This allows the algorithm to retain the optimal solution.

For the MATAP in this paper, because it is a discrete problem, we have to design a discrete mutation operation to replace the traditional mutation operation. The adapted operator of the classical DE algorithm is that the weighted difference vectors of two vectors are added to the third vector, as shown in Equation (7).

\[
v_{j,g} = x_{r_{1},g} + F * (x_{r_{2},g} - x_{r_{3},g})
\]

On the basis of not altering the core idea, we used a mutation mode as shown in Equation (8).

\[
v_{j,g} = x_{r_{1},g} \circ F \otimes (x_{r_{2},g} - x_{r_{3},g})
\]

\( g \) represents the evolutionary generation; \( r_{1} \neq r_{2} \neq r_{3} \neq j, j = 1, 2, \ldots, NP \), make sure the three variables are different variables, as is shown in lines 23 - 34 in pseudo-code of Algorithm 1; \( F \) is the mutation rate. Equation (8) contains two parts of operation. The first part is about the “\( \circ \)” operation, as illustrated below:

\[
\delta_{j,g} = F \otimes (x_{r_{2},g} - x_{r_{3},g})
\]

\[
= (x_{r_{2},g} - x_{r_{3},g}) \text{sgn}(\max(0, F - \text{rand}(0, 1)))
\]

\[
= \begin{cases} 
 x_{r_{2},g} - x_{r_{3},g} & \text{rand} < F \\
 0 & \text{others}
\end{cases}
\]

In Equation (9), \( \delta \) as an intermediate variable with the next “\( \otimes \)” operation, \( \delta = [\delta_{j,1,g}, \delta_{j,2,g}, \ldots, \delta_{j,n,g}] \). “sgn”, such that a number greater than 0 goes to 1 and a number less than 0 goes to \(-1\), 0 goes to 0.
The procedure is shown in the Figure 3.

You how to use our method to make row from 1 to 10 at this time. With an example, we’ll show as follows:

\[ p_{1,g} = x_{1,g} \oplus \delta_{1,g} \iff p_{1,n,g} = x_{1,n,g} \oplus \delta_{1,n,g} = \mod((x_{1,n,g} + \delta_{1,n,g} + N - 1), N) + 1 \]  

Equation (10) shows our idea of the “\( \oplus \)" operation. The main idea is to modify the solution variable by a modulo operation, so that we can prevent solution variable overflow from taking a range of values. To demonstrate the procedure, we will then show a concrete example of a mutation operation, as shown in Figure 2 (\( F = 0.5 \)). Take the first column as an example. First perform the “\( \oplus \)" operation, because \( \text{rand}(0,1) \) generates \( 0.2 < F = 0.5 \). \( \delta_{1,1,g} = x_{1,2,1,g} = -3 \). Then perform the “\( \ominus \)" operation, \( p_{1,1,g} = \mod((3 + (-3) + 10 - 1), 10) + 1 = 10 \).

\section{Crossover Operation}

As shown in the concrete example at the end of Section IV-C, \( p_j \) does not satisfy the criterion of not having repetition in each row from 1 to 10 at this time. With an example, we’ll show you how to use our method to make \( p_j \) satisfy the criterion. The procedure is shown in the Figure 3.

Assuming \( p_j = [12 \ 27 \ 15 \ 84 \ 25 \ 4] \), it is obvious that 2, 4, 5 are duplicates in the \( p_j \). To generate a random sequence by using the rand algorithm, because there are two 2, the random sequence is \([1 2] \), we save the first 2 and delete the second 2. In the same way, do the same for others. By replacing deleted places with 0 \( p_j \) is converted to \([10 27 100 80 54] \). Compared with the base variable \([123456789] \), \( p_j \) lacks \([369] \). Replace 0 in \( p_j \) with \([369] \) in a random sequence. Because the sequence is \([693] \), the new \( p_j \) is \([10 27 1689354] \), as illustrated in Figure 3.

\section{Update Speed and Position Variable}

The particles in the D-DEPSO algorithm have position and velocity variables. The position of one particle represents a task allocation scheme for multi-agents. The position variable is generated by the speed variable. The fitness of particles represents the expected loss of the task allocation scheme. Thus, Equation 1 is selected as the fitness function of the algorithm. A greater particle fitness corresponds to a lower expected loss of the task-allocation scheme that the particle represents.

Every time a particle-update operation is completed, the optimal particle is updated. The process is as follows: The fitness of the current position of a particle is calculated. If the fitness of \( x \) is less than the fitness of \( p_{best} \), which represents the known optimal position of \( x \), then \( p_{best} \) is replaced with the fitness of \( x \). Similarly, if \( p_{best} \) is lower than \( g_{best} \), which is the fitness of the global optimal position, then \( g_{best} \) is replaced with \( p_{best} \).

Traditional PSO is proposed by Kennedy and Eberhart in 1995 [26]. Speed-update formulation in this paper is shown in Equation 11:

\[ v(t + 1) = v(t) + c_1 \text{rand}(p - x) + c_2 \text{rand}(g - x) \]  

Y Shi and R Eberhart present PSO with inertia coefficient [27], it becomes standard PSO gradually. As is shown in below:

\[ v(t + 1) = wv(t) + c_1 \text{rand}(p - x) + c_2 \text{rand}(g - x) \]  

Based on Equation 12, we present a new speed-update formulation, as shown in Equation 13. In this equation, we define the operation “\( R \)”. The calculation results in parentheses are first handled by a boundary condition that removes the number of less than zero or more than ten, and then all the calculation results are rounded up to the next integer. Because new speed variable \( v(t + 1) \) does not meet the criterion of not have repetition in each row from 1 to 10, we handle it using the same method that presents in Section IV-D. \( w = 0.6 \), \( c_1 = 1.2 \), \( c_2 = 1.2 \) are the default values. Assume \( v(t) = [7968132540] \), \( x = [5946813710] \), \( p = [10271689354] \), \( g = [67910154328] \), first \( \text{rand} = 0.6 \) and second \( \text{rand} = 0.4 \). Before “\( R \)" operation, variable = \([8.28 -0.6 8.16 2.52 -4.2 8.76 -1.8 -5.04 10.32] \), by the “\( R \)" operation, \( v(t + 1) = [80830960010] \). Then we execute the crossover operation presented in Section IV-D.
FIGURE 4. Position-update operation.

A new \( v(t + 1) = [2 1 8 3 7 9 6 5 4 10] \) variable is created.

\[
v(t + 1) = R(wv(t) + c_1 \text{rand}(p - x) + c_2 \text{rand}(g - x))
\]

(13)

The position-update operation updates the original position variable through the recombination of the two variables \( x \) and \( v \). For the method of recombination, there are too many types of recombination operators [28]. A new approach was presented to the recombination operator based on the core idea of exacting a part of each variable of \( x \) and \( v \), respectively. Through this method, a new \( x \) variable was obtained. As shown in Figure 4, the procedure is as follows:

By generate random number 4, first four columns of \( x \) was extracted. It’s obvious that we need to extract 6 columns of \( v(t) \). Then, a new \( x \) variable was got by combining the two parts. The new \( x \) variable absolutely doesn’t satisfy the criterion of not having repetition in each row from 1 to 10. We also need to apply the crossover operation that is presented in Section IV-D. Then a new \( x \) variable that satisfies all the constraints is obtained.

The next experiments demonstrate the advanced and rational nature of the framework of D-DEPSO. Through this framework, D-DEPSO has a faster convergence speed and a better global value.

V. EXPERIMENTAL RESULT ANALYSIS

A. EXPERIMENTAL ANALYSIS OF ALGORITHM

Because of the time-critical nature of the epidemic, multi-agent task allocation problem in the epidemic scenario that has rarely been studied. Due to the large-scale and static nature of multi-agent task allocation problems in the epidemic scenario, no other algorithms are suitable for this particular scenario. So we compare the D-DEPSO with discrete DE, discrete PSO, improved discrete DE (IDE) [29], improved discrete PSO (IPSO) [30] and improved discrete genetic algorithm (IGA) [31] to illustrate the superiority of the proposed method.

To demonstrate and analyze the performance of the proposed D-DEPSO algorithm applied to an epidemic scenario, several experiments are implemented: (1) comparison of discrete DE, discrete PSO, improved discrete DE, improved discrete PSO, improved discrete GA and D-DEPSO algorithms on the performance of MATAP of the different scales and the different NP; (2) experiments on the performance of the D-DEPSO algorithm in MATAP of the same scales but with different population numbers (NP). In this paper, we assume that the parameters such as the value of an object, the probability of a robot \( i \) completing task \( j \) on time, and so on are definite.

All experiments are executed in the environment as follows: 64-bit Windows 11 21H2; 3.20 GHz AMD Ryzen 7 5800H with Radeon Graphics CPU; 16G Memory; programming environment: MATLAB R2020b.

In the experiments, the D-DEPSO algorithm was compared with discrete DE, discrete PSO, improved DE, improved PSO and improved GA algorithms in different dimensions and different NP. Matrices of \( v_k \) and \( p_{ij} \) are randomly generated, \( v_k \) presents value of executed task, every number in matrix of \( v_k \) is a random number between \([0, 100]\); \( p_{ij} \) presents probability of robot \( i \) completing task \( j \) on time, every number in matrix of \( p_{ij} \) is a random number between \([0, 1]\). For each comparison experiment of the same dimension and different NP, once \( v_k \) and \( p_{ij} \) are determined, they will not be changed. The parameters of the six algorithms involved in the experiments are set as follows (all parameters are optimal settings obtained by experiments):

1. DE/IDE
   (1) For the initial mutation rate, \( F = 0.4 \).
   (2) For the initial crossover rate, \( CR = 0.8 \).
2. PSO/IPSO
   (1) For the initial inertial coefficient, \( w = 0.8 \).
   (2) For the initial cognitive factor, \( c_1 = 1.2 \).
   (3) For the initial social factor, \( c_2 = 1.2 \).
3. IGA
   (1) For the weight coefficient, \( w = 0.15 \).
4. D-DEPSO
   (1) For the initial inertial coefficient, \( w = 0.8 \).
   (2) For the initial mutation rate, \( f = 0.8 \).
   (3) For the initial cognitive factor, \( c_1 = 1.2 \).
   (4) For the initial social factor, \( c_2 = 1.2 \).

Because of the calculating flexibility of high dimension. The maximum number of iterations of all three algorithms is set to 200 for all 10 and 20 dimensions, and the maximum number of iterations of all three algorithms is set to 500 for all 50 and 100 dimensions.

As shown in the first row in Figure 5. To illustrate the application of the above relative work in an epidemic scenario, we conduct some experiments in 10 dimensions based on simulated data that is generated by a random function. Ten dimensions means that there are ten tasks waiting to be matched to ten agents. In this case, there are three medicine distribution tasks, two medical material handling tasks, four periodic disinfection tasks, and one long-distance measurement of body temperature task. Every robot can execute any task.

Six algorithms, discrete DE, discrete PSO, improved DE, improved PSO, improved GA and D-DEPSO, were implemented for this case at 50, 100, and 200 NP. As you can see in the first row of Figure 5, there are the comparison plots...
of DE, PSO, IDE, IPSO, IGA and D-DEPSO algorithms in ten dimensions for the NP of 50, 100, and 200, respectively. It’s clear that D-DEPSO obtains optimal values not only for a small population but also for a large population. With the number of population increasing, D-DEPSO gets a better solution. But IGA does not differ significantly at NP of 100 or 200.

When the NP is 50, D-DEPSO still has a clear advantage. But when the NP is 100, IDE has a fastest convergence speed. When the NP is 200, IPSO has a faster convergence speed and the result of IGA almost catches up with D-DEPSO. As a whole, D-DEPSO has the best performance in this case.

As you can see in the second row of Figure 5, we conducted some experiments to compare the losses of six algorithms in twenty dimensions for the NP of 50, 100, and 200. There is almost the same trend as in ten dimensions. D-DEPSO can obtain an optimal solution in this scenario, but IGA has a faster convergence speed when NP are 50 and 200. When NP is 100, IDE and IPSO has a faster convergence speed. But D-DEPSO has the absolute advantage of solution in twenty dimensions.

For fifty-dimensional MATAP in an epidemic scenario, we have conducted simulation experiments on 50 tasks allocated to 50 robots in this part. Because of the increase in the number of tasks, we need a longer iteration period. The experimental results are presented in the third row of Figure 5. IPSO and IGA have started to show excellent performance. Whether NP is 50, 100 or 200, IPSO and IGA have almost equally fast convergence speed. But the D-DEPSO algorithm based on PSO has the better solution. The NP is bigger, so the optimal solution is better.

Due to one-hundred-dimensional MATAP has a more complex computation, we not only increase the number of iterations but also set the number of populations from 50, 100, 200 to 100, 200, 300. Experimental results have shown in the last row of Figure 5 that D-DEPSO has an absolute advantage in searching for a global optimal solution. For the high dimension MATAP in the epidemic scenario, D-DEPSO has a faster convergence speed and the lowest losses.

Because of the complexity of the task under the epidemic, we compare 10, 20, 50, and 100 dimensions in Figure 5. We can know that IPSO and IGA algorithms have fast convergence speed in the early stages of algorithm operation. For the low dimension, IDE can search for the sub-optimal solution fast and has a fast convergence speed. But it is easy to fall into the local optimal solution. The ability to search for global optimal solutions in IPSO, IGA and D-DEPSO is much better than in DE and IDE. In the cases of three medicine distribution tasks, two medical material handling tasks, four periodic disinfection tasks, and a long distance measurement of body temperature task. D-DEPSO can obtain the optimal solution [4,10,1,6,7,9,5,2,8,3] when “NP” is set to 100. Meanwhile, D-DEPSO can generate a minimum loss for the task allocation in the epidemic scenario. And for the 100 dimensions, contrast to the other five algorithms D-DEPSO has a definite advantage.

B. ANALYSIS OF STATISTICAL DATA
To facilitate our setting of the “NP” parameter, we test the effect of different populations on D-DEPSO’s handing MATAP in the same dimension. We conducted four experiments in 10, 20, 50, and 100 dimensions, respectively. The results are shown in Figure 6. The optimal solution generated when NP is 100 and 200 is almost the same. It means that after NP increases to 100, continuing to increase NP has little effect on the MATAP of 10 and 20 dimensions. For the fifty dimensions MATAP, D-DEPSO generates an optimal solution when NP is set at 200. For the one hundred-dimensional MATAP, the optimal solution generated when NP is 200 and 300 is almost the same. So the parameter setting of NP is set to the maximum of 200.

According to the above experiments, we get a result about the “NP” parameter setting. For the task allocation problems of 10 and 20 dimensions, “NP” set to 100 task allocation problems will have an optimal solution. However, for the higher dimensional task allocation problems of 50 and 100, the “NP” parameter setting of 200 is more appropriate.

In this part, the statistical data of discrete DE, discrete PSO, improved discrete DE, improved discrete PSO, improved discrete GA and D-DEPSO was summarized. Here are some conditions to clarify. The “NP” parameter is set using the experimental results above. From every dimension, mean value, standard deviation, and average time were obtained.
TABLE 3. Statistical data of DE, PSO, IDE, IPSO, IGA and D-DEPSO for different dimensions.

| Dim  | 10       | 20       | 50       | 100      |
|------|----------|----------|----------|----------|
| DE   | Loss min(k) | 39.0483  | 195.8512 | 657.6980 | 1646.8528|
|      | X ± S(k)  | 40.036±0.900 | 199.213±3.824 | 663.894±4.736 | 1651.169±4.962|
|      | Time avg(s)| 0.9811 | 1.9001 | 3.9566 | 13.968  |
| IDE  | Loss min(k) | 38.5648  | 186.6498 | 651.0314 | 1603.1858|
|      | X ± S(k)  | 39.7615±0.845 | 184.649±3.531 | 656.515±5.045 | 1611.5284±5.6482|
|      | Time avg(s)| 0.786 | 1.0709 | 3.4122 | 14.9648 |
| PSO  | Loss min(k) | 42.5667  | 168.3369 | 612.6988 | 1618.5760|
|      | X ± S(k)  | 43.714±0.929 | 172.879±4.651 | 621.694±6.652 | 1630.224±6.002|
|      | Time avg(s)| 0.8141 | 1.8214 | 4.3458 | 15.256  |
| IPSO | Loss min(k) | 40.6495  | 166.4562 | 596.1305 | 1618.1528|
|      | X ± S(k)  | 42.6781±0.8781 | 170.524±4.258 | 617.224±6.201 | 1628.548±6.1584|
|      | Time avg(s)| 0.9181 | 1.7458 | 4.1159 | 15.7856 |
| IGA  | Loss min(k) | 33.6195  | 172.1388 | 592.8468 | 1576.3252|
|      | X ± S(k)  | 33.0051±0.8975 | 176.482±3.168 | 609.468±4.861 | 1598.6482±4.841|
|      | Time avg(s)| 0.8715 | 1.7901 | 4.0648 | 14.4655 |
| D-DEPSO | Loss min(k) | 36.2665  | 142.6051 | 589.1729 | 1554.0243|
|      | X ± S(k)  | 36.121±0.631 | 147.006±3.385 | 594.226±4.07 | 1572.001±4.631|
|      | Time avg(s)| 1.0768 | 1.8864 | 3.5724 | 13.0047 |

FIGURE 6. Comparison of D-DEPSO algorithms’ losses in same dimension and different NP.

from 100 sets of data. The statistical data of all experimental results have been shown in Table 3.

Here are three types of data: the minimum loss, “X ± S”, and the average time for each algorithm, respectively. The unit of the minimum loss and X ± S are k, 1k means one thousand RMB economic losses. The unit of the average time is s, it means second. Bolded characters are the optimal values for each parameter. Next, we illustrate the Table 3 from ten, twenty, fifty, and one hundred dimensions, respectively.

In the ten-dimensional statistical data, IGA has the minimum loss and minimum mean in all six algorithms. Meanwhile, IDE has the shortest average time, and the average time of D-DEPSO is almost the same as that of DE. In comparison to discrete DE, discrete PSO, IDE, IPSO and IGA, D-DEPSO has a smaller standard deviation. It means that the statistical data of D-DEPSO for ten dimensions fluctuates less.

Compared to ten dimensions, it has a similar trend for twenty dimensions. D-DEPSO generates the smallest loss and obtains the minimum mean, but its average time is the almost longest. In contrast, IDE has the shortest average time. The smallest standard deviation is generated by IGA.

For the fifty-dimensional statistics, their trend is the same as the twenty-dimensional statistics. Minimum loss, minimum mean, and minimum standard deviation are all obtained by D-DEPSO. Of course, the average time of D-DEPSO has improved. As opposed to IDE, which has the shortest average time. IGA has the minimum standard deviation, it means statistics of IGA fluctuate less.

The trend begins to change in one-hundred dimensional statistical data. D-DEPSO not only obtains the minimum loss and mean but also produces the smallest standard deviation and average time at this time. The values of mean and standard deviation illustrate the result of D-DEPSO fluctuating less. D-DEPSO has the best performance in one-hundred dimensional experiments.

C. ANALYSIS OF ACTUAL TASK ALLOCATION

1) TEN DIMENSIONS MATAP

For the ten dimensions of MATAP in the epidemic scenario, (Vi of each task is a specific value obtained from our evaluation), D-DEPSO generates an optimal solution
[3,10,2,8,7,9,1,6,4,5] when “NP” is 50, the optimal solution is [4,10,1,6,7,9,5,2,8,3] when “NP” is 100, and the optimal solution is [4,10,1,6,7,9,5,2,8,3] when “NP” is 200. As a result of the conclusions in the previous Section V-B, we know that the “NP” must be set to 100 to produce the optimal solution for the 10 dimensional task allocation problem. So we choose the optimal solution when “NP” is set to 100 for analysis. The optimal solution is [4,10,1,6,7,9,5,2,8,3] when NP is 100 and the matrix transformed by the optimal solution is shown in Matrix 14. For the matrix transformed by the optimal solution, there are some things we must interpret: the first column to third column represent medicine distribution tasks; the fourth column and fifth column represent medical material handling tasks; the sixth column to ninth column represent periodic disinfection tasks; and the tenth column represents long-distance measurement of body temperature tasks. It means that robots No.3, No.8 and No.10 are in charge of medicine distribution tasks; robots No.1 and No.7 are in charge of medical material handling tasks; robots No.4, No.5, No.6 and No.9 are in charge of periodic disinfection tasks; and robot No.2 is in charge of long-distance measurement of body temperature tasks.

\[
Y = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\] (14)

2) TWENTY DIMENSIONS MATAP

In this part, a twenty-dimensional task allocation in the epidemic scenario was illustrated. From the conclusion of the previous Section V-B, for twenty-dimensional MATAP, “NP” set to 100 is most suitable. So we obtained the optimal solution generated by D-DEPSO in twenty dimensions and 100 “NP”. The optimal solution is [13,10,15,20,9,16,18,7,12,4,11,8,6,3,19,1,5,17,14,2], where the number of integer locations represents the number of the robots and the integer represents the number of the tasks. The meaning of the optimal solution is shown in Figure 7. We can know the optimal task allocation is that robots No.16, No.20, No.14, No.10, No.17, and No.13 perform medicine distribution tasks; robots No.8, No.12, No.5, and No.2 execute medical material handling tasks; robots No.11, No.9, No.1, No.19, No.3, No.6, No.18, and No.7 implement periodic disinfection tasks; and robots No.15 and No.4 perform long-distance measurement of body temperature tasks.

Actual experiments show that D-DEPSO has good performance in twenty dimensions; it not only finds the optimal solution but also has a fast convergence speed.

Limited by the length of the article, 50-dimensional and 100-dimensional MATAP in the epidemic scenario are not specifically described here.

VI. CONCLUSION

This paper proposed a discrete hybrid algorithm named D-DEPSO to handle multi-agent task allocation problems based on a task allocation model and a meta-heuristic algorithm. Based on the constraints of the epidemic scenario, a multi-agent task allocation strategy was proposed, and the task allocation problem in the epidemic scenario was introduced and defined using the mathematical model. D-DEPSO is used to minimize the loss of task allocation. D-DEPSO improves the diversity of the population by mutation operation, and it combines the discrete PSO algorithm to improve the global searching ability. The results of experiments that compare D-DEPSO with the other five algorithms demonstrate that the D-DEPSO algorithm obtains optimal solutions in different dimensions, and the running speed of D-DEPSO is faster than the discrete DE, the discrete PSO, IDE, IPSO and IGA in 100 dimensions. Thus, we think that D-DEPSO has better performance in higher dimensions. And it shows that D-DEPSO handles large-scale task allocations with significant advantages. Due to the large scale task in the epidemic scenario, D-DEPSO will be able to obtain lower losses and more diverse solutions compared to the other five algorithms.

However, this paper only considered single-objective optimization. In the future, we will concentrate on the multi-objective optimization of multi-agent task allocation problems in epidemic scenarios. More complex task scenarios will be considered in future work.

REFERENCES

[1] Y. Sun, Z. Ji, Q. Qi, and H. Ma, “Bipartite consensus of multi-agent systems with intermittent interaction,” IEEE Access, vol. 7, pp. 130300–130311, 2019.
[2] M. Roscia, M. Longo, and G. C. Lazaroiu, “Smart city by multi-agent systems,” in Proc. Int. Conf. Renew. Energy Res. Appl. (ICRERA), Oct. 2013, pp. 371–376.
[3] L. Yin, J. Luo, and H. Luo, “Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing,” IEEE Trans. Ind. Informat., vol. 14, no. 10, pp. 4712–4721, Oct. 2018.
[4] R. Bahnemann, D. Schindler, M. Kamel, R. Siegwart, and J. Nieto, “A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects,” in Proc. IEEE Int. Symp. Saf. Secur. Rescue Robot. (SSRR), Oct. 2017, pp. 123–128.
[5] X. Jin, M. Imiti, and H. Abdullah, “A cooperative multi-agent system simulation model for urban traffic intelligent control,” in Proc. Summer Comput. Simul. Conf., 2007, pp. 953–958.
[6] L.-J. Pan, P. Y. Yao, and P. Sun, “Distributed task allocation method of manned/unmanned combat agents,” Syst. Eng. Electron., vol. 35, no. 2, pp. 310–316, 2013.

[7] G.-Z. Yang, B. J. Nelson, R. R. Murphy, H. Choset, H. Christensen, S. H. Collins, P. Dario, K. Goldberg, K. Ikatka, N. Jacobstein, D. Kragic, R. H. Taylor, and M. McNeill, “Combating COVID-19—The role of robotics in managing public health and infectious diseases,” Sci. Robot., vol. 5, no. 40, Mar. 2020, Art. no. eabb5589.

[8] Y. Jiang, Y. Wang, J. Yi, Z. Miao, L. Liu, H. Zhong, and H. Zhang, “Research on intelligent robot systems for emergency prevention and control of major pandemics,” Scientia Sinica Informatica, vol. 50, no. 7, pp. 1069–1090, Jul. 2020.

[9] S. Jeon, J. Lee, and J. Kim, “Multi-robot task allocation for real-time hospital logistics,” in Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC), Oct. 2017, pp. 2465–2470.

[10] H. S. Stone, “Multiprocessor scheduling with the aid of network flow algorithms,” IEEE Trans. Softw. Eng., vol. SE-3, no. 1, pp. 85–93, Jan. 1977.

[11] K. W. Tindell, A. Burns, and A. J. Wellings, “Allocating hard real-time tasks: An NP-hard problem made easy,” Real-Time Syst., vol. 4, no. 2, pp. 145–165, Jun. 1992.

[12] P. C. Cheeseman, B. Kanefsky, and W. M. Taylor, “Where the really hard problems are,” in Proc. IJCAI, vol. 91, 1991, pp. 331–337.

[13] J. D. Ullman, “NP-complete scheduling problems,” J. Comput. Syst. Sci., vol. 10, no. 3, pp. 384–393, 1975.

[14] A. Ernst, H. Jiang, and M. Krishnamoorthy, “Exact solutions to task allocation problems,” Manag. Sci., vol. 52, no. 10, pp. 1634–1646, Oct. 2006.

[15] C. Zhang, J. Chen, and B. Xin, “Distributed metric differential evolution with the synergy of Lamarckian and Baldwinian learning,” Appl. Soft Comput., vol. 13, no. 5, pp. 2947–2959, 2013.

[16] B. Xin, J. Chen, J. Zhang, H. Fang, and Z.-H. Peng, “Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy,” IEEE Trans. Syst., Man, Cybern., C, vol. 42, no. 5, pp. 744–767, Sep. 2012.

[17] J. Schwarzrock, I. Zacarias, A. L. Bazzan, R. Q. D. A. Fernandes, L. H. Moreira, and E. P. D. Freitas, “Solving task allocation problem in multi unmanned aerial vehicles systems using swarm intelligence,” Eng. Appl. Artif. Intell., vol. 72, pp. 10–20, Jun. 2018.

[18] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, “Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm,” IEEE Access, vol. 8, pp. 19306–19318, 2020.

[19] W. Lu, W. Zhiliang, H. Siquan, and L. Lei, “Ant colony optimization for task allocation in multi-agent systems,” Commun., China, vol. 10, no. 3, pp. 125–132, Mar. 2013.

[20] D. Jing, C. Jian, and S. Min, “Cooperative task assignment for heterogeneous multi-UAVs based on differential evolution algorithm,” in Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst. Nov. 2009, pp. 163–167.

[21] M. Nouri, A. Bekrar, and A. Jemai, “An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem,” J. Intell. Manuf., vol. 29, no. 3, pp. 603–615, Mar. 2018.

[22] F. Zhang, “Intelligent task allocation method based on improved QPSO in multi-agent system,” J. Ambient Intell. Humanized Comput., vol. 11, no. 2, pp. 655–662, Feb. 2020.

[23] J. C. Amorim, V. Alves, and E. P. De Freitas, “Assessing a swarm-GAP based solution for the task allocation problem in dynamic scenarios,” Exp. Syst. Appl., vol. 152, Aug. 2020, Art. no. 113437.

[24] C. M. Zhang and J. C. B. Xin, “A discrete differential evolution algorithm for the weapon target allocation problem,” Trans. Beijing Inst. Technol., vol. 34, pp. 289–293, Jan. 2014.

[25] R. Storn and K. Price, “Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces,” J. Global Optim., vol. 11, no. 4, pp. 341–359, 1997.

[26] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proc. Int. Conf. Neural Nets., vol. 4, 1995, pp. 1942–1948.

[27] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in Proc. IEEE Int. Conf. Evol. Comput. Proc., IEEE World Congr. Comput. Intell., May 1998, pp. 69–73.

[28] A. Banks, J. Vincent, and C. Anyakoha, “A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multiteria and constrained optimization, and indicative applications,” Natural Comput., vol. 7, no. 1, pp. 109–124, Mar. 2008.

[29] M. K. Sallam, S. M. Elsayed, R. K. Chakraborty, and M. J. Ryan, “Improved multi-operator differential evolution algorithm for solving unconstrained problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jul. 2020, pp. 1–8.

[30] L. Tong, B. Du, R. Liu, and L. Zhang, “An improved multiobjective discrete particle swarm optimization for hyperspectral endmember extraction,” IEEE Trans. Geosci. Remote Sens., vol. 57, no. 10, pp. 7872–7882, Oct. 2019.

[31] X. Wu, Y. Yin, L. Xu, X. Wu, F. Meng, and R. Zhen, “MULTI-UAV task allocation based on improved genetic algorithm,” IEEE Access, vol. 9, pp. 100369–100379, 2021.