J Syst Sci Complex (2017) 30: 357–377

Scenario-Based Stochastic Resource Allocation with Uncertain Probability Parameters*

FAN Guimei · HUANG Haijun

DOI: 10.1007/s11424-017-6178-5
Received: 16 August 2016 / Revised: 10 November 2016

Abstract The stochastic resource allocation (SRA) problem is an extensive class of combinatorial optimization problems widely existing in complex systems such as communication networks and unmanned systems. In SRA, the ability of a resource to complete a task is described by certain probability, and the objective is to maximize the reward by appropriately assigning available resources to different tasks. This paper is aimed at an important branch of SRA, that is, stochastic SRA (SSRA) for which the probability for resources to complete tasks is also uncertain. Firstly, a general SSRA model with multiple independent uncertain parameters (GSSRA-MIUP) is built to formulate the problem. Then, a scenario-based reformulation which can address multi-source uncertainties is proposed to facilitate the problem-solving process. Secondly, in view of the superiority of the differential evolution algorithm in real-valued optimization, a discrete version of this algorithm was originally proposed and further combined with a specialized local search to create an efficient hybrid optimizer. The hybrid algorithm is compared with the discrete differential evolution algorithm, a pure random sampling method, as well as a restart local search method. Experimental results show that the proposed hybrid optimizer has obvious advantages in solving GSSRA-MIUP problems.

Keywords Discrete differential evolution, scenario-based reformulation, stochastic resource allocation.

1 Introduction

Stochastic resource allocation (SRA) is an extensive class of combinatorial optimization problems originating from many complex systems, e.g., radio resource allocation in communication networks[1–4], network topology management[5] and relay resources management of wireless networks[6], portfolio management[7], circuit design[8], mission planning in unmanned

FAN Guimei · HUANG Haijun
School of Economics and Management, Beihang University, Beijing 100191*, China.*
Email: fgmBeihang@163.com.
*This research was supported by the National Natural Science Foundation of China under Grant No. 71361130 011.
◇ *This paper was recommended for publication by Editor SUN Jian.*

systems[9], resource allocation in multi-agent systems[10], reliability optimization in complex systems[11], and weapon-target assignment in defense-oriented research fields[12–15].

A distinct feature of SRA is reflected by the fact that SRA solutions depend on the probabilities of stochastic elements or/and events. For example, the reliability of a complex system relies on those of all its components, and the combating effectiveness of a defense system relies on the performance of all involved weapons against incoming targets. According to literature survey, various types of SRA are usually characterized as a deterministic optimization problem with fixed probability parameters[13] or a stochastic one with certain probability distribution of some key elements[2]. For example, in most research regarding weapon-target assignment (WTA) which can be regarded as a typical example of SRA problems, the kill probabilities of weapons against targets are assumed to be known a priori as constant[12–15]. In contrast, Xu, et al.[2] established a stochastic optimization model for the resource allocation of feedback in clustered wireless mesh networks by taking into account the probability distribution of signal-to-noise ratios. Obviously, as uncertainty widely exists in practice, the stochastic optimization (programming) model may be preferred as a rational choice for SRA in most cases. In the following, the two types of SRA models mentioned above are differentiated by deterministic SRA (DSRA) and stochastic SRA (SSRA), respectively. Generally speaking, the SRA in complex systems usually involves uncertainty from different sources. For example, as for WTA problems, the kill probabilities of different weapons against targets usually follow independent distributions[16]. In this sense, SRA is generally a stochastic optimization problem with multiple uncertain parameters. For addressing this kind of stochastic SRA problems, the idea of sampling-based simulation is employed to generate different combinations of uncertain parameters which lead to various scenarios[17]. Each scenario corresponds to one possibility of the uncertain SRA problem.

In order to solve SSRA problems efficiently, a powerful optimization algorithm, namely, differential evolution (DE), is employed and redesigned in this paper so as to fit the problem structure of SSRA. DE is a population-based intelligent optimization method which uses the difference between individuals to perturb the population so as to explore the whole search space. DE also adopts greedy selection to evolve solutions in pursuit of global optima[18, 19]. Classical DE adopts an encoding scheme based on floating numbers. So, it is mainly used to solve continuous-valued optimization problems. It has the virtue of high reliability, strong robustness, and desirable optimization performance. DE holds only a few parameters, so it is very easy to use and realize. Due to these facts, DE has been a hotspot in evolutionary computation research[18].

In view of the excellent performance of DE in continuous-valued optimization, many researchers are trying to build its discrete versions to solve combinatorial optimization problems. Forward-backward transformation (FBT) and relative position indexing (RPI) are two representatives for designing discrete DE (DDE)[20]. However, like many other DDEs, FBT and RPI were proposed primarily to solve permutation problems in which solutions are encoded as permutation of many elements[20].

The motivation of this paper is to design a DE-based efficient optimization algorithm to

solve a class of general SSRA problems. The originality of the paper is reflected in the following aspects:

1) A general SSRA model with multiple independent uncertain parameters (GSSRA-MIUP) is established originally. The GSSRA-MIUP model can characterize a class of uncertain programming problems regarding the resource allocation in complex systems.

2) A scenario-based reformulation of GSSRA-MIUP is proposed to address the issue of multi-source uncertainty incurred by multiple independent uncertain parameters. Uncertain parameters are randomly sampled according to their probability distribution in order to generate different scenarios for any GSSRA-MIUP instance. Then, the collection of scenarios leads to a computable formulation of the GSSRA-MIUP model. The reformulation facilitates a desirable tradeoff between optimality and computational cost.

3) A novel hybrid discrete differential evolution (HDDE) algorithm is proposed to solve GSSRA-MIUP problems. A specifically designed discrete DE is designed and further enhanced by local search, giving birth to the hybrid optimizer HDDE. Comparative experiments validate the superiority of HDDE against several other candidates for solving GSSRA-MIUP problems.

The remaining of the paper is structured as follows. Section 2 gives two formulations of GSSRA-MIUP, including expected value model and robust optimization model. To address multi-source uncertainty, GSSRA-MIUP is reformulated by generating a collection of scenarios through random sampling of uncertain parameters. In Section 3, a discrete differential evolution algorithm is proposed and further enhanced by local search methods to solve GSSRA-MIUP problems. Section 4 presents the computational experiments and results, and makes a performance comparison of several problem-solvers for GSSRA-MIUP. Section 5 concludes the paper.

2 Formulation of GSSRA-MIUP and Scenario-Based Reformulation

2.1 Basic Formulations of GSSRA-MIUP

In the SSRA, there are R resources to be assigned to T tasks. The effect of assigning a resource to a task is reflected by the probability of completing the task by the selected resource. Naturally, the results of performing all the tasks vary with allocation schemes. A reward will be gained for each task. The objective here is to maximize the total reward by appropriately assigning resources to tasks. The total reward function can be expressed as follows:

$$f(X,\xi)=\sum_{j=1}^{T} v_j\left[1-\prod_{i=1}^{R}(1-\xi_{ij})^{x_{ij}}\right], \tag{1}$$

$$x_{ij}\in\{0,1\},\quad \xi_{ij}\sim\mathcal{S}_{ij}[0,1],\quad \forall i\in I_R=\{1,2,\cdots,R\},\quad \forall j\in I_T=\{1,2,\cdots,T\}, \tag{2}$$

where $X=[x_{ij}]_{R\times T}$ is a decision matrix, and $x_{ij}=1$ means that the ith resource is assigned to the jth task ($x_{ij}=0$, otherwise); $\xi=[\xi_{ij}]_{R\times T}$ is a matrix of stochastic variables which are independently distributed within the interval $[0,1]$ usually with non-identical distributions $\mathcal{S}_{ij}[0,1]$ ($i\in I_R, j\in I_T$), I_R and I_T are the index sets of the resources and tasks, respectively; ξ_{ij} reflects the probability that the ith resource, once assigned, will complete the jth task;

and $v_j (j \in I_T)$ is the reward for completing the jth task. The first mathematical model for GSSRA-MIUP is formulated as follows:

$$\text{P1}: \quad \max_{X} \quad E[f(X,\xi)] \tag{3}$$

$$\text{s.t.} \quad \sum_{j=1}^{T} x_{ij} \le m_i, \quad \forall i \in I_R, \tag{4}$$

$$\sum_{i=1}^{R} x_{ij} \le n_j, \quad \forall j \in I_T, \tag{5}$$

where $E[\cdot]$ denotes the expectation operator, m_i is the maximal number of tasks to which the ith resource can be assigned, and n_j is the maximal number of resources which can be assigned to the jth task. Inequality (4) means that the ith resource can be assigned to at most m_i tasks. Inequality (5) means that the jth task can be assigned to at most n_j resources.

The second model is differentiated from the above formulation P1 only in term of the objective function:

$$\text{P2}: \quad \max_{X} \quad \min_{\xi} \quad f(X,\xi) \tag{6}$$

$$\text{s.t.} \quad (4) \text{ and } (5).$$

The model P1 is the expected value model for GSSRA-MIUP. In contrast, P2 is a robust optimization model. P1 is a rational choice for a risk-taker while P2 is usually preferred by a risk-averse decision-maker. Other models such as chance-constrained model and dependent-chance model can also be adopted for modeling GSSRA-MIUP[17]. Whether the formulation P1 or P2 is adopted, the GSSRA-MIUP is obviously an uncertain nonlinear 0-1 programming problem. Due to the existence of multiple random variables in GSSRA-MIUP, it is usually very hard to find a straightforward solution to the problem. In the sequel, reformulations will be provided to facilitate computations and resolution methods.

2.2 Scenario-Based Reformulation

The multi-source uncertainty incurred by multiple independent uncertain parameters as shown in (3) and (6) impedes the computation of objective functions. The expected value shown in (3) and the minimum of the total reward function with respect to the uncertainty ξ shown in (6) are cumbersome, if not possible, to calculate accurately. Here, a scenario-based reformulation of GSSRA-MIUP is proposed to address the issue.

Firstly, uncertain parameters are randomly sampled according to their probability distribution in order to generate different scenarios for any GSSRA-MIUP instance. Each combination of uncertain parameters leads to one scenario, representing one possibility of the uncertainty. Then, the collection of the various scenarios generated by random sampling will be incorporated into the GSSRA-MIUP model (P1 or P2) for approximately calculating $E[f(X,\xi)]$ or $\min_{\xi} f(X,\xi)$. This approach is also called simulation-based method in the literature as the

Monte Carlo simulation is the core means to realize random sampling[17, 21]. The reformulation is presented as follows:

$$\text{P3}: \quad \max_{X} \quad \frac{1}{N}\sum_{k=1}^{N} f(X, \xi^{(k)}) \tag{7}$$
$$\text{s.t.} \quad (4) \text{ and } (5),$$
$$\text{P4}: \quad \max_{X} \quad \min_{k=1,2,\cdots,N} \quad f(X, \xi^{(k)}) \tag{8}$$
$$\text{s.t.} \quad (4) \text{ and } (5),$$

where $\xi^{(k)}$ denotes the matrix of the specific values of all uncertain parameters in the kth scenario, and N is the total number of scenarios generated by random sampling.

Remark As possible scenarios may be infinite, a limited number of samplings may only lead to an approximate representation of the uncertainty. However, since the sampling process follows the distribution of all uncertain parameters, these randomly generated scenarios are adequate in a statistical sense if its quantity is sufficiently large. Therefore, the above reformulation facilitates a desirable tradeoff between optimality and computational cost. Based on the reformulation, various optimization methods can be employed or designed to get near-optimal or even optimal solutions to GSSRA-MIUP.

3 Problem-Solver: Hybrid Discrete Differential Evolution

3.1 Basic Operations of the New Discrete Differential Evolution Algorithm

The proposed discrete differential evolution (DDE) algorithm will be applied to the solution X directly. In other words, the binary representation of solutions to GSSRA-MIUP is adopted here. The DDE maintains a population of PS individuals (solutions) simultaneously, denoted by $Pop = \{X_1, X_2, \cdots, X_{PS}\}$. In the following, the differential mutation, crossover, and selection operations of the DDE will be presented in sequence. These operations inherit the main features of the corresponding operations of the original DE variant for real-valued optimization, namely DE/rand/1/bin[18].

1) Binary differential mutation (BDM) operation

The BDM operation with respect to (w.r.t.) the ith individual X_i $(i = 1, 2, \cdots, PS)$ can be described as follows:

$$V_i = X_{r_1} \oplus [F_i \odot (X_{r_2} \ominus X_{r_3})], \tag{9}$$

where $V_i = [v^i_{j,k}]_{R\times T}$ is the mutant individual w.r.t. X_i; X_{r_1}, X_{r_2}, and X_{r_3} are three individuals (solutions) randomly selected from the current DE population with r_1, r_2, $r_3 \in \{1, 2, \cdots, PS\}$ and $r_1 \neq r_2 \neq r_3 \neq i$; $F_i = [f^i_{j,k}]_{R\times T} = [\text{rand} < \alpha]_{R\times T}$ where α is a decisive factor whose function is similar to the scaling factor in DE/rand/1/bin, rand$= [r_{j,k}]_{R\times T}$ is an $R \times T$ dimensional matrix with each element being a random number uniformly distributed in the interval $(0, 1)$ (if $r_{j,k} < \alpha$, then $f^i_{j,k} = 1$; otherwise $f^i_{j,k} = 0$); the symbol '$\oplus$' denotes the elementwise

exclusive OR (XOR) operator, '$\odot$' denotes the elementwise AND operation of two matrices, and '$\ominus$' denotes the elementwise logical complement of XOR (i.e., XNOR).

Remark The differential mutation is the most important and distinct feature which distinguishes DE from the genetic algorithms. It is a crucial component which endows DE with excellent search performance in continuous solution space. The key idea of the differential mutation is to use the scaled difference of randomly paired solutions to perturb another solution so as to generate a new one. The above discrete BDM operator is aimed at mimicking the operations of its continuous counterpart 'DE/rand/1' (denoted by $V_i = X_{r_1} + F_i \cdot (X_{r_2} - X_{r_3})$) which has proved to be very successful in lots of applications[18]. It can be observed from (9) that the BDM operator shares the same feature with 'DE/rand/1'.

2) Binomial crossover

The binomial crossover here is very similar to that in DE/rand/1/bin. The operation will generate a trial individual $U_i = [u^i_{j,k}]_{R\times T}$ by combining the mutant individual $V_i = [v^i_{j,k}]_{R\times T}$ with the current individual $X_i = [x^i_{j,k}]_{R\times T}$:

$$u^i_{j,k} = \begin{cases} v^i_{j,k}, & \text{if } \mathrm{rand}^i_{j,k} < CR; \\ x^i_{j,k}, & \text{otherwise}, \end{cases} \qquad \begin{array}{l} i = 1, 2, \cdots, R; \quad j = 1, 2, \cdots, T, \end{array} \tag{10}$$

where $\mathrm{rand}^i_{j,k}$ is a random number uniformly distributed in the interval $(0, 1)$, and $CR \in (0, 1)$ is the so-called crossover rate.

3) Constraint handling: Repairing and improvement

As the BDM operator and the binomial crossover cannot guarantee the feasibility of the generated solutions, that is, the constraints (4) and (5) may be violated. Therefore, it is necessary to design specific constraint handling methods to repair those infeasible solutions. Besides, it can be seen from Equation (1) that, assigning more resources will lead to a better objective value. In this sense, if no constraints are violated, it is rational to add as many ones as possible into a solution, which can be utilized to improve the quality of solutions. Also, it is evident that the violation of constraints (4) and (5) is due to the excessive ones in certain rows and columns, respectively. Based on the above facts, we propose the following constraint handling method (see Algorithm 1). In the process, excessive ones will be removed while more ones will also be added only if they do not cause violation of any constraints. The constraint handling method will be applied to each trial solution U_i before calculating its objective value.

4) 1 versus 1 tournament selection

The trial individual U_i after repairing and improvement will compete against the current individual X_i, and the winner will survive to the next generation.

$$X_i = \begin{cases} U_i, & \text{if } F(U_i) \geq F(X_i); \\ X_i, & \text{otherwise}, \end{cases} \tag{11}$$

where $F(X)$ is the objective value of the solution X.

Algorithm 1 Constraint Handling (Repairing and Improvement)

Input X
Denote by SR_i the sum of all elements in the ith row of X; Denote by SC_j the sum of all elements in the jth column of X.
Compute all SR_i $(i = 1, 2, \cdots, R)$; IF $SR_i > m_i$ THEN $RCV_i = 1$ ELSE $RCV_i = 0$ END //$RCV_i = 1$ indicates the ith row sum constraint is violated
Compute all SC_j $(j = 1, 2, \cdots, T)$; IF $SC_j > n_j$ THEN $CCV_j = 1$ ELSE $CCV_j = 0$ END //$CCV_j = 1$ indicates the jth column sum constraint is violated
R_{ind} = randperm(R), C_{ind} = randperm(T) // randperm(n) is a random permutation of the integers from 1 to n; the two random permutations are used in the following to remove 'irrational' biases towards the constraints corresponding to anterior rows or anterior columns.
FOR $i = 1$ to R // First, handle the constraints w.r.t. each row, i.e., the constraints shown by (4).
    $p = R_{ind}(i)$;
    IF $RCV_p == 1$
        Randomly select $(SR_p - m_p)$ 1-valued elements from the pth row of X, and change them to zeros. Add as many ones as possible into the columns corresponding to the changed elements, if no constraints are violated.
    ELSE
        Add as many ones as possible into the pth row of X, if no constraints are violated.
    END
END FOR
FOR $j = 1$ to T // Second, handle the constraints w.r.t. each column, i.e., the constraints shown by (5).
    $q = C_{ind}(j)$;
    IF $CCV_q == 1$
        Randomly select $(SC_q - n_q)$ 1-valued elements from the qth column of X, and change them to zeros. Add as many ones as possible into the columns corresponding to the changed elements, if no constraints are violated.
    ELSE
        Add as many ones as possible into the qth column of X, if no constraints are violated.
    END IF
END FOR
Output X

3.2 Local Search

Local search (LS) is often integrated into population-based metaheuristics (e.g., DE) to enhance its exploitation. Local search is built on certain neighborhood of solutions. Here, the following two kinds of neighborhood are proposed for GSSRA-MIUP.

Definition 1 Task Neighborhood. The task neighborhood of a solution is the set of all solutions which can be reached by changing the assigned resources w.r.t. one certain task.

For example, $X_1 = \left[\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{smallmatrix}\right]$ is a neighbor of the solution $X_2 = \left[\begin{smallmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{smallmatrix}\right]$ w.r.t. the second task (2nd column).

Definition 2 Resource Neighborhood. The resource neighborhood of a solution is the set of all solutions which can be reached by changing the assigned tasks w.r.t. one certain resource.

For example, $X_1 = \left[\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{smallmatrix}\right]$ is a neighbor of the solution $X_3 = \left[\begin{smallmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right]$ w.r.t. the first resource (1st row).

In GSSRA-MIUP, it is more important to assign tasks to resources rather than the reverse since tasks usually have to be addressed but not all resources have to be used. Therefore, the LS operations are defined on task neighborhood. Here, two LS operations are proposed:

1) Single Task Neighborhood Local Search (STNLS)

For a selected task, a part of the resources assigned to it will be replaced by the same amount of other resources which have not been depleted. A neighboring solution which is generated in this way will be compared with the current solution, and the better one will be chosen as the current solution. The local search continues with the current solution until a predefined number of LS operations is reached.

Assume, for example, that each resource can be assigned to at most two tasks. The solution $X_1 = \left[\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{smallmatrix}\right]$ can be changed to $X_4 = \left[\begin{smallmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{smallmatrix}\right]$ by an STNLS operation w.r.t. the second task.

2) Multiple Task Neigborhoods Local Search (MTNLS)

For a selected task, a part of the resources assigned to it will be replaced by the same amount of other resources which have been depleted (fully assigned to other tasks). In this case, the replacement may violate the row constraint regarding the related resources. In this sense, one of the tasks previously assigned to related resources has to be reassigned to another resource. Besides, the reassignment of affected tasks is also subjected to the same issue mentioned above. Therefore, the operation will bring the changes of assignment in multiple task neighborhoods. A neighboring solution will be generated in this way until no constraints are violated. The new solution will be compared with the current solution, and the better one will be chosen as the current solution. The local search continues with the current solution until a predefined number of LS operations is reached.

Assume, for example, that each resource can be assigned to at most two tasks. For the solution $X_1 = \left[\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{smallmatrix}\right]$, if the first task is reassigned to the first resource, then the tasks which were previously assigned to the first resource, that is the second or the third task, will have to be reassigned to the second resource, giving birth to $X_5 = \left[\begin{smallmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{smallmatrix}\right]$ or $X_6 = \left[\begin{smallmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right]$.

In each generation, the above two LS operations will be applied to the individuals in the population according to a predefined selection probability p_{LS}. Besides, STNLS and MTNLS

Algorithm 2 Local Search

Input X
FOR $k = 1$ to n_{LS} //n_{LS} is the allowable number of local search operations.
  Randomly select one task by the probability vector $\boldsymbol{P}_{stn} = [P_{stn}(1), P_{stn}(2), \cdots, P_{stn}(T)]$;
  Denote the task by t'.
  // The probability for choosing the ith task is $P_{stn}(i) = \frac{\sum_{j \in I_T} v_i p_{ij}}{\sum_{i \in I_R} \sum_{j \in I_T} v_i p_{ij}}$,
    where p_{ij} is a specific value of ξ_{ij} which is dependent on specific GSSRA-MIUP models.
  Denote by $\mathcal{R}_{t'}$ the set of the resources previously assigned to t'. Randomly select one resource from $\mathcal{R}_{t'}$ by the probability vector $\boldsymbol{P}_{rem1} = [P_{rem1}(1), P_{rem1}(2), \cdots, P_{rem1}(L_1)]$ where $L_1 = sizeof(\mathcal{R}_{t'})$. Remove it from $\mathcal{R}_{t'}$. Randomly assign as many other tasks as possible to the resource.
  // The probability for choosing the mth resource in $\mathcal{R}_{t'}$ is
$$P_{rem1}(m) = \frac{\frac{1}{p_{ira(m),t'}}}{\sum_{ira(m) \in R_{t'}} \frac{1}{p_{ira(m),t'}}},$$
  where $ira(m)$ is the index of the mth resource in $\mathcal{R}_{t'}$.
  Denote by $\overline{\mathcal{R}}_{t'}$ the set of the resources which previously were not assigned to t'. Randomly assign a new resource to t' by $\boldsymbol{P}_{res1} = [P_{res1}(1), P_{res1}(2), \cdots, P_{res1}(L_2)]$ where $L_2 = sizeof(\overline{\mathcal{R}}_{t'})$. Denote the new resource by r'. Denote the resulting solution by Y.
  // The probability for choosing the mth resource which was not assigned to t' is
$$P_{res1}(m) = \frac{p_{irn(m),t'}}{\sum_{irn(m) \in \overline{\mathcal{R}}_{t'}} p_{irn(m),t'}},$$
  where $irn(m)$ is the index of the mth resource in $\overline{\mathcal{R}}_{t'}$.
  WHILE 1
    IF Y does not violate any constraint
      Replace X by Y if $F(Y) \geq F(X)$; //Y is generated by an STNLS operation
      BREAK;
    ELSE
      Denote by $\mathcal{T}_{r'}$ the set of the tasks previously assigned to r'. Randomly select one task from $\mathcal{T}_{r'}$ by $\boldsymbol{P}_{mtn} = [P_{mtn}(1), P_{mtn}(2), \cdots, P_{mtn}(L_3)](L_3 = sizeof(\mathcal{T}_{r'}))$. Denote the chosen task by t^*. Remove r' from the set of the resources assigned to t^*.
      // The probability for choosing the nth task in $\mathcal{T}_{r'}$ is
$$P_{mtn}(n) = \frac{\frac{1}{\sum_{j \in I_T} v_{ita(n)} p_{ij}}}{\sum_{ita(n) \in \mathcal{T}_{r'}} \frac{1}{\sum_{j \in I_T} v_{ita(n)} p_{ij}}},$$
      where $ita(n)$ is the index of the nth task in $\mathcal{T}_{r'}$.
      Denote by $\overline{\mathcal{R}}_{t^*}$ the set of the resources which previously were not assigned to t^*.
      IF $\overline{\mathcal{R}}_{t^*}$ is not empty
        Randomly assign a new resource to t^* by $\boldsymbol{P}_{res2} = [P_{res2}(1), P_{res2}(2), \cdots, P_{res2}(L_4)]$ $(L_4 = sizeof(\overline{\mathcal{R}}_{t^*}))$. Reuse r' to denote the new resource.
        //The probability for choosing the nth resource in $\overline{\mathcal{R}}_{t^*}$ is $P_{res2}(n) = \frac{1}{L_4}$, where $irn(n)$ is the index of the nth resource in $\overline{\mathcal{R}}_{t^*}$.
      END IF
      Reuse Y to denote the resulting solution. //This is an MTNLS operation.
    END IF
  END WHILE
END FOR
Output X

will be randomly implemented within the same procedure as shown in Algorithm 2. A search bias towards potentially better solutions is made by using four parameters to control the local search, including $\boldsymbol{P}_{stn}$, $\boldsymbol{P}_{mtn}$, $\boldsymbol{P}_{res1}$, and $\boldsymbol{P}_{rem1}$ (see Algorithm 2).

Remark 1 It is easy to see from Algorithm 2 that the setting of $\boldsymbol{P}_{stn}$ and $\boldsymbol{P}_{res1}$ prefers choosing the combination of tasks and resources with larger potential rewards. In contrast, the setting of $\boldsymbol{P}_{rem1}$ and $\boldsymbol{P}_{mtn}$ prefers deleting the combination of tasks and resources with relatively smaller potential rewards.

Remark 2 The setting of the parameter $\boldsymbol{P}_{res2}$ has no bias towards any resource so as to avoid frequent cyclic exchange of resources with higher potential rewards among several tasks. This is because frequent cyclic exchange of the same resources may repeatedly generate the same solutions.

Algorithm 3 Generation of Initial Population

Input PS // Population size
FOR $k = 1$ to PS
  $X_k = \boldsymbol{O}_{R\times T}$; // Zero matrix
  $ID_R = \text{randperm}(R)$;
  $s_R = \text{zeros}(1, R)$;
  FOR $i = 1$ to R
    $ID_T = \text{randperm}(T)$;
    FOR $j = 1$ to T
      IF $X_k(ID_R(i), ID_T(j)) := 1$ does not violate any constraint
        $X_k(ID_R(i), ID_T(j)) = 1$;
        $s_R(ID_R(i), ID_T(j)) = s_R(ID_R(i), ID_T(j)) + 1$;
        IF $s_R(ID_R(i), ID_T(j)) \geq m_{ID_R(i)}$
          BREAK;
        END IF
      END IF
    END FOR
  END FOR
END FOR
Output $X_1, X_2, \cdots, X_{PS}$

3.3 Hybrid DDE for GSSRA-MIUP

1) Procedure of the Hybrid DDE

The local search and the DDE proposed in the preceding context will be combined into a hybrid algorithm, named hybrid DDE (HDDE). After each selection operation in DDE, the local search will be implemented on the current solution by a probability P_{LS}. To gain a better tradeoff between exploration and exploitation[22], the regulation parameter is set as $P_{LS} = NFE_{acc}/NFE_{\max}$ where NFE_{acc} is the accumulated number of function evaluations

and $NFE_{\max}$ is the maximal number of function evaluations. Obviously, local search will be more frequently used when the search progresses. The procedure of the algorithm HDDE is presented as follows:

Step 1 Population Initialization.

Step 2 Evaluate initial solutions and record the so-far-best solution.

Step 3 Apply local search to initial solutions according to P_{LS}, and update the so-far-best solution (see Algorithm 2).

Step 4 Main Loop

For each solution X_i $(i = 1, 2, \cdots, PS)$, do the following:

Step 4.1 Apply the BDM operation to obtain the mutant solution V_i (see Equation (9)).

Step 4.2 Apply the binomial crossover operation to obtain the trial solution U_i (see Equation (10)).

Step 4.3 Apply the constraint handling method to repair U_i (see Algorithm 1).

Step 4.4 Apply the tournament selection to update X_i (see Equation (11)). Update the so-far-best solution.

Step 4.5 Apply the local search to X_i according to P_{LS} (see Algorithm 2). Update the so-far-best solution.

Step 4.6 If the termination criterion is satisfied, stop the algorithm and output the so-far-best solution; otherwise, go to Step 4.1.

2) Population Initialization

Due to the constraints shown in (4) and (5), it is necessary to design deliberate generation methods to avoid a large amount of infeasible solutions especially in the case of $m_i << T$ and $n_j << R$ in which a feasible solution X will be a sparse matrix. The generation method for initial solutions is presented in Algorithm 3.

Table 1 Computational complexity

Operations	Time complexity	Space complexity
Initialization	$\mathcal{O}(T * R * PS)$	$\mathcal{O}(T * R * PS)$
Function Evaluation	$\mathcal{O}(T * R * N * \mathrm{NFE}_{\max})$	$\mathcal{O}(T * R)$
BDM	$\mathcal{O}(T * R * \mathrm{NFE}_{\max})$	$\mathcal{O}(T * R)$
Crossover	$\mathcal{O}(T * R * \mathrm{NFE}_{\max})$	$\mathcal{O}(T * R)$
Selection	$\mathcal{O}(T * R * \mathrm{NFE}_{\max})$	$\mathcal{O}(1)$
Constraint Handling	$\mathcal{O}(T * R * \mathrm{NFE}_{\max})$	$\mathcal{O}(T + R)$
Local Search	$\mathcal{O}(T * R * n_{LS} * \mathrm{NFE}_{\max})$	$\mathcal{O}(T + R)$

3.4 Computational Complexity

The time and space complexity of HDDE can be analyzed according to its procedure including seven operations as shown above. The result of the worst-case analysis w.r.t. each operation is presented in Table 1. In view of the fact $PS << \mathrm{NFE}_{\max}$, the total worst-case time and space complexity are $\mathcal{O}((N + n_{LS}) * \mathrm{NFE}_{\max} * T * R)$ and $\mathcal{O}(T * R * PS)$, respectively. If

the parameter setting in Subsection 4.2 is adopted, then the worst-case time complexity and the worst-case space complexity are $\mathcal{O}((T*R)^{2.5})$ and $\mathcal{O}((T*R)^{1.5})$, respectively.

4 Computational Experiments

4.1 GSSRA-MIUP Test Instances

In order to test the performance of the proposed HDDE for solving GSSRA-MIUP, ten instances with different problem scales, parameters, and probability distributions of ξ are constructed as follows:

$$\begin{aligned}
&\text{Instance1}: \ T=3, \ R=3; && \text{Instance2}: \ T=5, \ R=5;\\
&\text{Instance3}: \ T=10, \ R=10; && \text{Instance4}: \ T=10, \ R=20;\\
&\text{Instance5}: \ T=20, \ R=10; && \text{Instance6}: \ T=20, \ R=20;\\
&\text{Instance7}: \ T=20, \ R=50; && \text{Instance8}: \ T=50, \ R=20;\\
&\text{Instance9}: \ T=50, \ R=50; && \text{Instance10}: \ T=50, \ R=100.
\end{aligned}$$

For each instance, $v_i = 10 + 90 * \frac{i-1}{T-1}$ and $m_i = \lfloor 1 + 2 * \text{rand}_{1\times R} \rfloor$ for $i = 1, 2, \cdots, R$, $n_j = \lfloor 1 + 3 * \text{rand}_{1\times T} \rfloor$ for $j = 1, 2, \cdots, T$, and $\xi_{ij} \sim \mathcal{U}[\xi_{ij}^-, \xi_{ij}^+]$ $(i = 1, 2, \cdots, R;\ j = 1, 2, \cdots, T)$ where $\mathcal{U}[\xi_{ij}^-, \xi_{ij}^+]$ means a uniform distribution in the interval $[\xi_{ij}^-, \xi_{ij}^+]$, $\xi_{ij}^- = \min\{\text{rand}_{ij}^1, \text{rand}_{ij}^2\}$, $\xi_{ij}^+ = \max\{\text{rand}_{ij}^1, \text{rand}_{ij}^2\}$, rand_{ij}^1 and rand_{ij}^2 are two random numbers uniformly distributed in the interval $[0, 1]$.

4.2 Parameter Setting

There are totally seven parameters in HDDE, including PS, α, CR, P_{LS}, n_{LS}, $\text{NFE}_{\max}$, and p_{ij}. The setting of these parameters is shown as follows:

Population size: $PS = \lfloor 4\sqrt{R*T} \rfloor$.

Probability parameter for controlling BDM operations: $\alpha = 1 - 0.95 * \frac{\text{NFE}_{acc}}{\text{NFE}_{\max}}$.

Crossover rate: $CR = 0.9$.

Probability of applying local search operations: $P_{LS} = \frac{\text{NFE}_{acc}}{\text{NFE}_{\max}}$.

Number of the local search operations when applied: $n_{LS} = \lfloor 5\sqrt{R*T} \rfloor$.

Maximal number of function evaluations: $\text{NFE}_{\max} = 500 * R * T$.

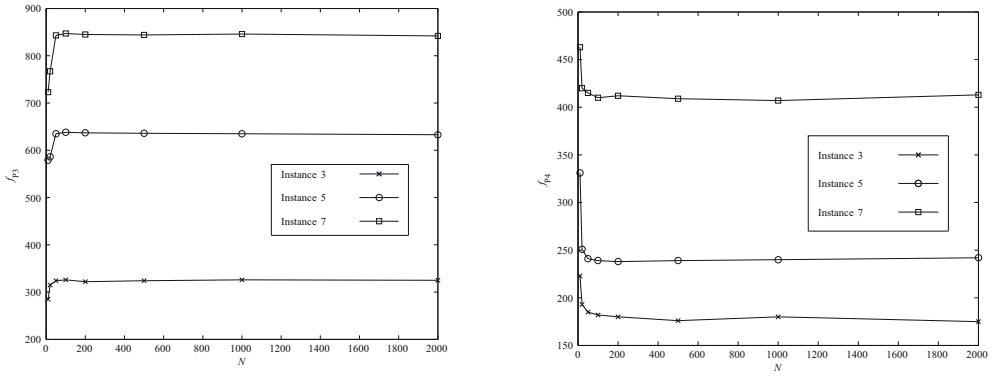
Probability parameter for controlling the search bias in local search: $p_{ij} = \text{mean}\{\xi_{ij}^{(k)}\}$ for the expected value model P3, and $p_{ij} = \min_k\{\xi_{ij}^{(k)}\}$ for the robust optimization model P4.

The setting of α and P_{LS} is beneficial to draw a tradeoff between exploration and exploitation in the search space[22]. PS, n_{LS}, and $\text{NFE}_{\max}$ vary with the scale of specific GSSRA-MIUP instances.

4.3 Sensitivity Test about the Number of Scenarios

Instances 3, 5 and 7 are used to make a sensitivity test w.r.t. the number of scenarios, i.e., the parameter N (see (7) and (8)). The settings $N = 10, 20, 50, 100, 200, 500, 1000$ and 2000 are considered. Besides, both the expected value model P3 and the robust optimization model P4 are included. In each case, HDDE was run independently 50 times and the best result is

selected as data for conducting the analysis. The discovered best objective values as a function of the parameter N are shown in Figure 1. It can be observed that the discovered objective values in different cases, whether the expected value model or the robust optimization model is adopted, have no obvious change when $N > 50$. According to the result, $N = 50$ is the best choice since a smaller N will obviously reduce computation cost. Therefore, $N = 50$ will be used in the following computational experiments.



(a) Expected value model (b) Robust optimization model

Figure 1 Sensitivity analysis about the number of scenarios N. f_{P3} and f_{P4} are the objective function values obtained by HDDE among 50 independent runs corresponding to the expected value model P3 and the robust optimization model P4, respectively

4.4 Comprehensive Comparison

In this section, comparative experiments will be made to evaluate the performance of the proposed HDDE algorithm against the basic DDE without local search, a pure random sampling method (PRS), as well as a restart local search (RLS). The basic DDE is the discrete differential evolution algorithm proposed in Section 3.1. PRS is de facto Algorithm 3 described above which maintains a very large population whose size is equal to the allowable number of function evaluations (i.e., $PS = \text{NFE}_{\max}$). RLS can be regarded as a combination of Algorithm 2 (initialization) and Algorithm 3 (local search). In RLS, the local search is always triggered by the solutions generated by Algorithm 2 which produces only one solution (i.e., $PS = 1$) at a time. Every time the local search is completed, the Algorithm 2 will provide a new solution to it and restart the local search until the maximal number of function evaluations is reached.

The setting of $\text{NFE}_{\max}$ is kept the same for all competitors so that these algorithms will spend almost the same computational cost for a fair comparison. The setting of p_{ij} is also the same for all. DDE keeps the same setting for PS, α, CR as HDDE. RLS keeps the same setting for n_{LS} as HDDE. For each instance with the model P3 or P4, each algorithm is run 50 times independently. All the tests were implemented on a PC with Intel i7-5500U CPU (2.4GHz) and 8GB Internal Memories.

Table 2 Statistical results about the performance of the four algorithms with the expected value model

Ins.	HDDE	DDE	PRS	RLS
No.	(min, max, mean, std)	(min, max, mean, std)	(min, max, mean, std)	(min, max, mean, std)
1	**(217.3, 217.3, 217.3, 0.0)**	(217.3, 217.3, 217.3, 0.0)	(217.3, 217.3, 217.3, 0.0)	(217.3, 217.3, 217.3, 0.0)
2	**(312.6, 312.6, 312.6, 0.0)**	(312.6, 312.6, 312.6, 0.0)	(312.6, 312.6, 312.6, 0.0)	(312.6, 312.6, 312.6, 0.0)
3	**(349.9, 350.1, 349.9, 0.1)**	(340.2, 349.9, 344.3, 2.6)	(307.4, 328.2, 317.9, 5.9)	(335.5, 337.9, 336.9, 0.7)
4	**(559.9, 560.1, 560.0, 0.1)**	(552.8, 558.4, 556.2, 1.6)	(532.6, 545.9, 539.6, 3.6)	(553.2, 560.1, 556.3, 2.1)
5	**(639.0, 640.9, 640.0, 0.4)**	(636.5, 638.6, 637.5, 0.6)	(613.2, 626.4, 620.4, 4.0)	(630.5, 639.0, 634.7, 2.3)
6	**(720.4, 725.4, 723.0, 1.2)**	(705.3, 712.2, 709.3, 1.7)	(681.5, 706.3, 694.2, 7.5)	(710.7, 722.6, 717.6, 3.1)
7	**(842.9, 849.6, 845.6, 1.5)**	(817.4, 826.3, 821.5, 2.8)	(782.3, 802.5, 794.1, 5.8)	(831.9, 845.2, 838.0, 3.7)
8	**(2358, 2372, 2366, 2.9)**	(2242, 2261, 2251, 4.8)	(2037, 2253, 2151, 66.7)	(2254, 2319, 2288, 18.4)
9	**(2671, 2684, 2677, 2.9)**	(2568, 2611, 2589, 12.5)	(2475, 2539, 2504, 20.0)	(2568, 2632, 2599, 18.9)
10	**(2732, 2744, 2739, 3.0)**	(2688, 2713, 2700, 7.3)	(2593, 2663, 2628, 19.8)	(2692, 2736, 2713, 11.3)

Table 3 Statistical results about the performance of the four algorithms with the robust optimization model

Ins.	HDDE	DDE	PRS	RLS
No.	(min, max, mean, std)	(min, max, mean, std)	(min, max, mean, std)	(min, max, mean, std)
1	**(124.1, 124.1, 124.1, 0.0)**	(124.1, 124.1, 124.1, 0.0)	(124.1, 124.1, 124.1, 0.0)	(124.1, 124.1, 124.1, 0.0)
2	**(157.0, 157.0, 157.0, 0.0)**	(157.0, 157.0, 157.0, 0.0)	(157.0, 157.0, 157.0, 0.0)	(157.0, 157.0, 157.0, 0.0)
3	**(179.9, 180.1, 180.0, 0.1)**	(173.2, 175.5, 174.4, 0.7)	(162.3, 170.8, 164.6, 1.4)	(169.2, 179.9, 174.4, 3.0)
4	**(223.9, 224.1, 224.0, 0.1)**	(213.7, 219.5, 217.0, 1.7)	(201.2, 211.4, 206.0, 3.0)	(215.3, 221.6, 218.7, 1.7)
5	**(243.7, 246.2, 245.1, 0.5)**	(224.3, 231.8, 228.0, 2.2)	(213.8, 225.6, 219.9, 3.5)	(227.5, 239.2, 233.8, 3.3)
6	**(332.9, 339.3, 335.9, 1.3)**	(310.5, 325.4, 318.0, 4.6)	(301.9, 314.2, 307.8, 3.4)	(314.0, 326.5, 320.2, 3.7)
7	**(411.7, 418.0, 415.0, 1.4)**	(382.1, 402.7, 392.0, 6.4)	(361.3, 375.0, 368.8, 3.8)	(371.6, 406.3, 388.0, 9.5)
8	**(1626, 1645, 1633, 4.0)**	(1582, 1603, 1595, 5.5)	(1562, 1589, 1576, 6.8)	(1587, 1620, 1604, 9.9)
9	**(1978, 1990, 1985, 2.7)**	(1927, 1946, 1937, 5.4)	(1882, 1913, 1898, 8.5)	(1945, 1963, 1955, 5.1)
10	**(2003, 2021, 2011, 3.6)**	(1958, 1981, 1970, 6.7)	(1913, 1947, 1930, 10.3)	(1973, 1995, 1985, 5.7)

Table 4 The mean and standard deviation of the time cost about the four algorithms (unit: Second)

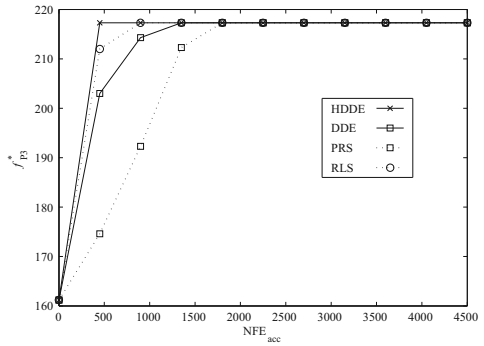
Ins.	HDDE		DDE		PRS		RLS	
No.	Model P3	Model P4	Model P3	Model P4	Model P3	Model P4	Model P3	Model P4
1	0.001±0.001	0.001±0.001	0.001±0.001	0.001±0.001	0.001±0.001	0.001±0.001	0.001±0.001	0.001±0.001
2	0.010±0.001	0.010±0.001	0.015±0.001	0.015±0.001	0.022±0.001	0.022±0.001	0.007±0.001	0.007±0.001
3	0.40±0.02	0.40±0.02	0.52±0.01	0.52±0.01	1.21±0.01	1.22±0.01	0.35±0.01	0.35±0.01
4	2.02±0.04	2.01±0.04	2.62±0.02	2.61±0.02	9.23±0.05	9.22±0.05	1.66±0.01	1.65±0.01
5	2.33±0.05	2.31±0.05	2.73±0.02	2.71±0.02	12.32±0.06	12.41±0.06	1.74±0.01	1.73±0.01
6	13.21±0.12	13.17±0.13	17.48±0.04	17.47±0.04	61.75±0.14	61.52±0.16	10.4±0.05	10.2±0.05
7	107.8±0.9	107.2±1.0	127.5±0.2	127.7±0.2	312.4±0.5	310.7±0.5	88.1±0.08	86.4±0.08
8	124.6±1.2	123.0±1.2	166.2±0.4	164.8±0.4	361.8±0.6	360.1±0.6	101.8±0.09	100.2±0.09
9	1037.1±2.0	1035.8±2.1	1241.0±2.4	1256.8±2.5	2917.4±2.4	2913.2±2.7	827.5±0.6	824.3±0.7
10	3263.3±5.4	3251.9±6.2	4069.2±4.3	4042.7±4.9	6432.3±4.5	6429.1±4.8	2415.2±0.8	2413.0±0.8

Statistical results are presented in Tables 2 and 3. It is obvious that HDDE outperforms DDE, PRS, and RLS in terms of all statistical indexes in all cases except for instances 1 and 2. As for instances 1 and 2, all competitors show the same performance, which is mainly because the two instances are very simple. However, as the scale of the instances increases, the superiority of HDDE over the other algorithms becomes more and more evident. The Wilcoxon rank sum test with 0.05 confidence level was conducted in each case. For any two algorithms A and B, when their performances are significantly different according to the hypothesis test, if the average result of A is better than that of B, then $h(A, B) = 1$; if the average result of A is worse than that of B, then $h(A, B) = -1$; otherwise, $h(A, B) = 0$. The test results are shown as follows, and each result includes the tests regarding the comparison of the corresponding algorithms in all the ten instances from instance 1 to instance 10. $\boldsymbol{h}_{P3}(\cdot, \cdot)$ corresponds to the test results about the expected value model (P3) while $\boldsymbol{h}_{P4}(\cdot, \cdot)$ corresponds to the test results about the robust optimization model (P4).

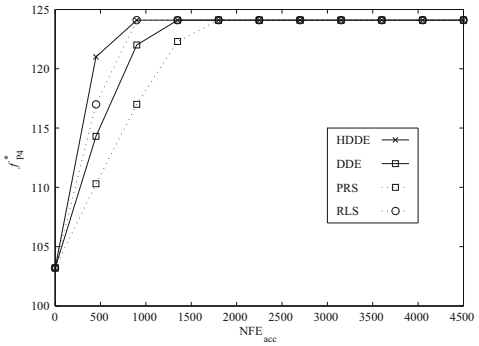
The hypothesis test results also agree with the intuitional judgement from the statistical results presented in Tables 2 and 3. The statistical results about the time cost of each algorithm are presented in Table 4. As the maximal number of function evaluations ($\mathrm{NFE}_{\max}$) is the same for all algorithms, the results can reflect their operational complexity. Obviously, PRS is the most time-consuming among the four algorithms while RLS holds the least time cost. HDDE has a larger cost than RLS while it saves much time as compared with DDE. The advantage of HDDE over DDE reflects the effectiveness of incorporating the local search into DDE to gain a tradeoff between exploration and exploitation[22]. DDE outperforms the pure random sampling (PRS) but loses to RLS, which also confirms the necessity of combining DDE with local search.

Table 5 Results of Wilcoxon rank sum test

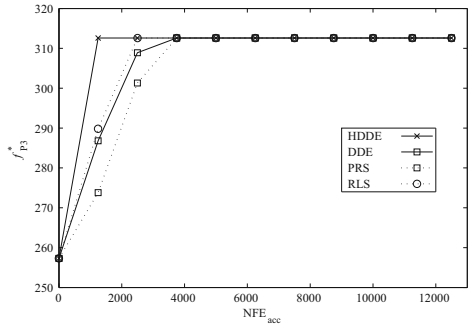
Comparison	Instance No.									
Index	1	2	3	4	5	6	7	8	9	10
$\boldsymbol{h}_{P3}$(HDDE, DDE)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P4}$(HDDE, DDE)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P3}$(HDDE, PRS)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P4}$(HDDE, PRS)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P3}$(HDDE, RLS)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P4}$(HDDE, RLS)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P3}$(DDE, PRS)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P4}$(DDE, PRS)	0	0	1	1	1	1	1	1	1	1
$\boldsymbol{h}_{P3}$(DDE, RLS)	0	0	1	0	1	−1	−1	−1	−1	−1
$\boldsymbol{h}_{P4}$(DDE, RLS)	0	0	0	−1	−1	−1	0	−1	−1	−1
$\boldsymbol{h}_{P3}$(PRS, RLS)	0	0	−1	−1	−1	−1	−1	−1	−1	−1
$\boldsymbol{h}_{P4}$(PRS, RLS)	0	0	−1	−1	−1	−1	−1	−1	−1	−1



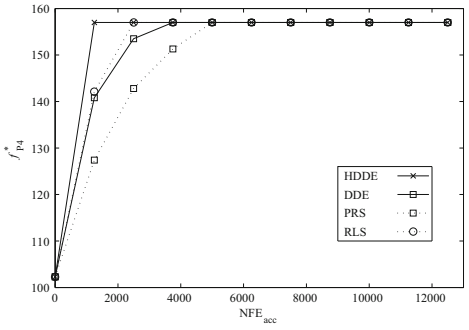
(a) Instance 1, P3



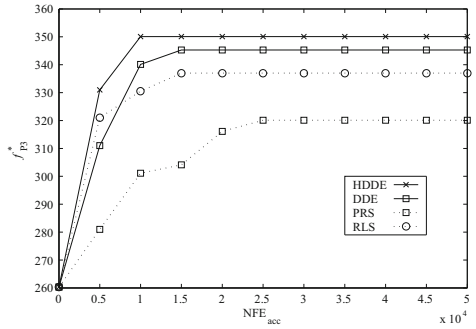
(b) Instance 1, P4



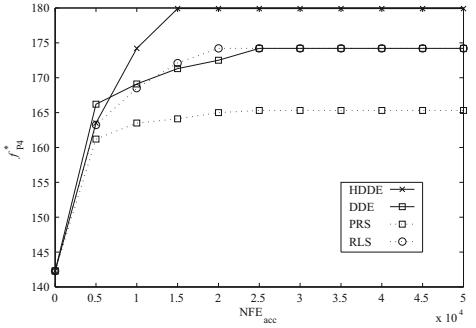
(c) Instance 2, P3



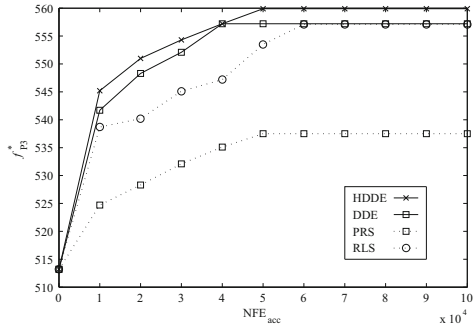
(d) Instance 2, P4



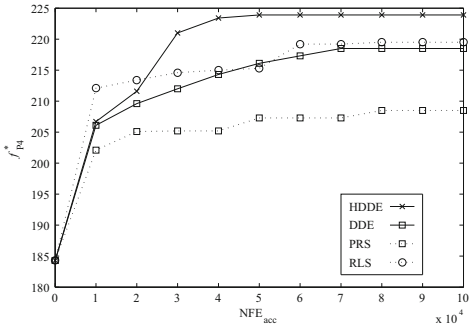
(e) Instance 3, P3



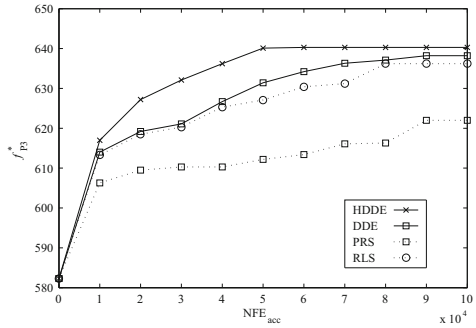
(f) Instance 3, P4



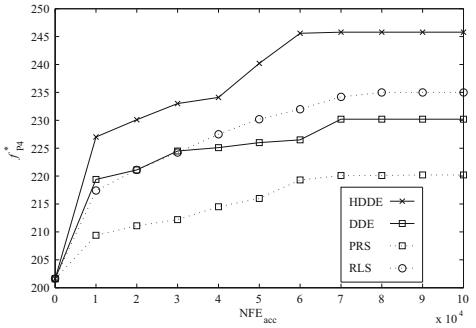
(g) Instance 4, P3



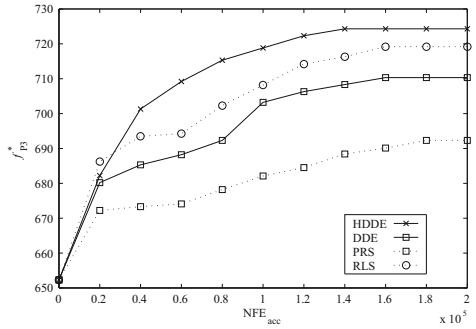
(h) Instance 4, P4



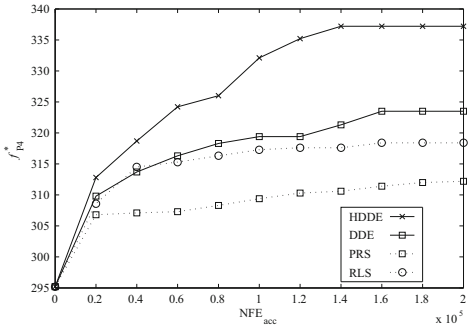
(i) Instance 5, P3



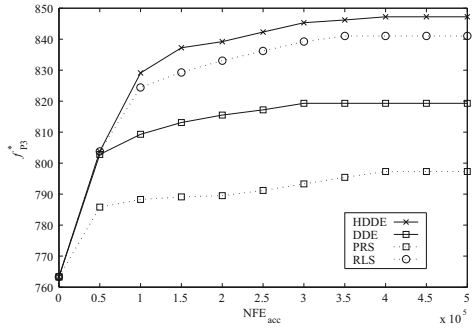
(j) Instance 5, P4



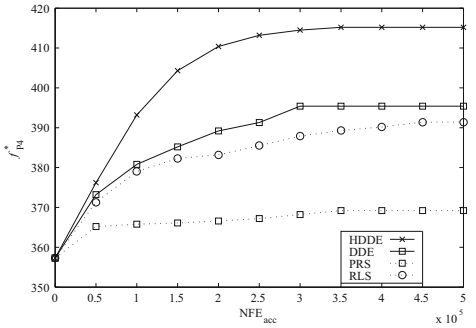
(k) Instance 6, P3



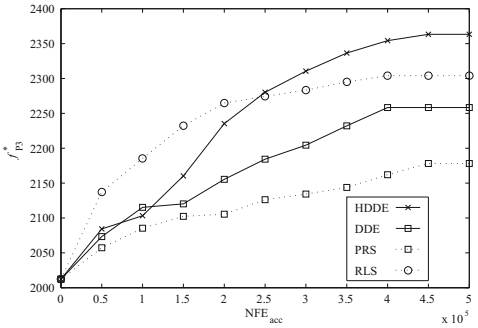
(l) Instance 6, P4



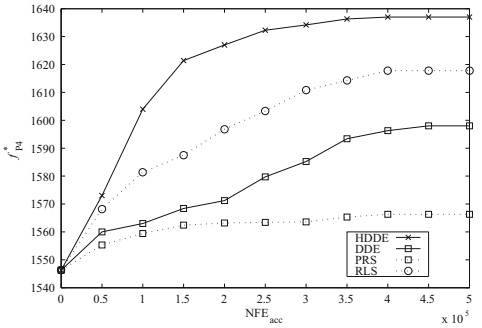
(m) Instance 7, P3



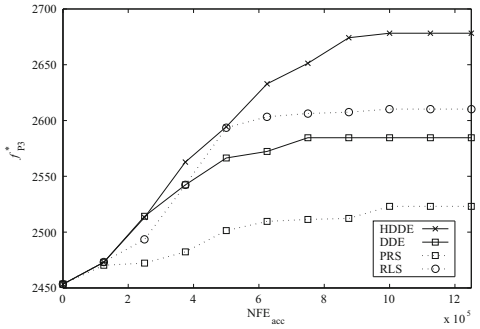
(n) Instance 7, P4



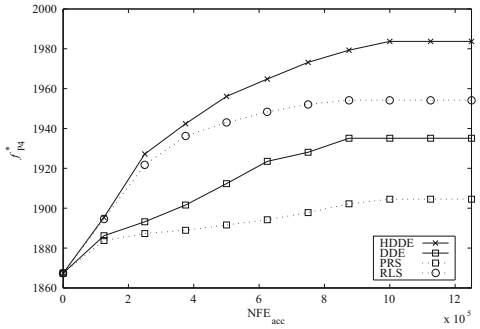
(o) Instance 8, P3



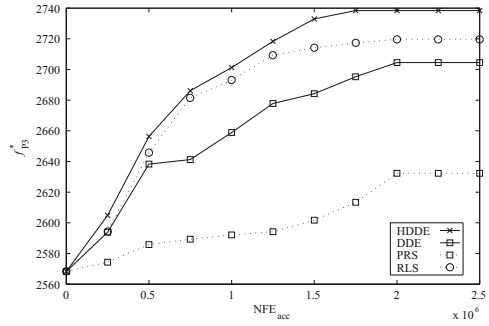
(p) Instance 8, P4



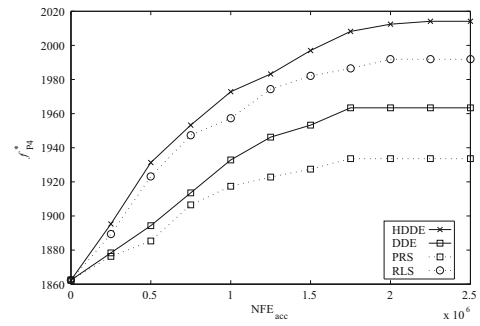
(q) Instance 9, P3



(r) Instance 9, P4



(s) Instance 10, P3



(t) Instance 10, P4

Figure 2 Convergence curves of four algorithms

The convergence process of the four algorithms in each case is presented in Figure 2. From these convergence curves, it can be observed that HDDE converges faster than the other three algorithms in almost all cases. Regarding Instance 3 (P4), HDDE converges a bit slower than RLS at the very early stage, however, the solution finally discovered by HDDE is far better than that by RLS. Similar results are also visible with regard to the comparison between HDDE and RLS in solving Instance 4 (P4), Instance 6 (P3), and Instance 8 (P3). Besides, RLS performs better than DDE in terms of convergence speed in most cases except for Instance 3 (P3), Instance 4 (P3), Instance 5 (P3), Instance 6 (P4), and Instance 7 (P4). In contrast, PRS is no doubt the slowest among the four algorithms since it does not use any heuristic information during the search, and lacks a proper balance between the exploration and exploitation of the search space. To sum up, HDDE is the best algorithm for GSSRA-MIUP among the four algorithms.

5 Conclusion

The GSSRA-MIUP problem represents a wide range of uncertain resource allocation problems involved in complex systems. A scenario-based reformulation is proposed to convert the original problem into a computable form. Based on the reformulation, a discrete differential

evolution algorithm was proposed and further combined with local search to create an efficient hybrid optimizer, namely HDDE. The performance of HDDE in solving GSSRA-MIUP problems is validated through a set of test instances. The results of comparative computational experiments show that HDDE not only can effectively solve GSSRA-MIUP problems but also has obvious advantages over the other three feasible problem-solvers.

References

[1] Saki H, Shojaeifard A, and Martini M G, Stochastic resource allocation for hybrid spectrum access OFDMA-based cognitive radios, *Proceedings of the IEEE International Conference on Communications* (*ICC*), London, England, June 8–12, 2015, 7750–7755.

[2] Xu L, Yang Y, and Li Y, Resource allocation of limited feedback in clustered wireless mesh networks, *Wirel. Person. Comm.*, 2014, **75**(2): 901–913.

[3] Li X, Shi Y, Wang X J, et al., Efficient link scheduling with joint power control and successive interference cancellation in wireless networks, *Science China Information Sciences*, 2016, **59**(12): No.122301.

[4] Wang X and Giannakis G B, Resource allocation for wireless multiuser OFDM networks, *IEEE Trans. Infor. Theor.*, 2011, **57**(7): 4359–4372.

[5] Wang L Y, Lin F, and Yin G, Network robustness depth and topology management of networked dynamic systems, *J. Syst. Sci. Complex.*, 2016, **29**(1): 1–21.

[6] Chu H Y, Xu P P, Wang W, and Yang C C, Joint relay selection and power control for robust cooperative multicast in mmWave WPANs, *Science China-Information Sciences*, 2016, **59**(8): No.082301.

[7] Yin L B and Han L Y, Risk management for international portfolios with basket options: A multi-stage stochastic programming approach, *Journal of Systems Science & Complexity*, 2015, **28**(6): 1279–1306.

[8] Li M H, Huang G M, Wu X L, et al., and Zhou D, A yield-enhanced global optimization methodology for analog circuit based on extreme value theory, *Science China-Information Sciences*, 2016, **59**(8): No.082401.

[9] Melouk S H, Fontem B A, Waymire E, et al., Stochastic resource allocation using a predictor-based heuristic for optimization via simulation, *Comput. Oper. Res.*, 2014, **46**: 38–48.

[10] Plamondon P and Chaib-Draa B, Stochastic resource allocation in multiagent environments: An approach based on distributed Q-values and bounded real-time dynamic programming, *Int. J. Artif. Intell. Tool.*, 2012, **21**(1): 1250003.

[11] Kuo W and Wan R, Recent advances in optimal reliability allocation, *IEEE Trans. Syst. Man, Cybern. A — Syst. Human.*, 2007, **37**(2): 143–156.

[12] Chen J, Xin B, Peng Z H, et al, Evolutionary decision-makings for dynamic weapon-target assignment problem, *Sci. China Ser. F: Inf. Sci.*, 2009, **52**(11): 2006–2018.

[13] Castanon D A and Wohletz J M, Model predictive control for stochastic resource allocation, *IEEE Trans. Autom. Contr.*, 2009, **54**(8): 1739–1750.

[14] Xin B, Chen J, Peng Z H, et al, An efficient rule-based constructive heuristic to solve dynamic

weapon-target assignment problem, *IEEE Trans. Syst. Man Cybern. A — Syst. Human.*, 2011, **41**(3): 598–606.

[15] Xin B, Chen J, Zhang J, Dou L H, and Peng Z H, Efficient decision-makings for dynamic weapon-target assignment by virtual permutation and tabu search heuristics, *IEEE Trans. Syst. Man Cybern. C — Appl. Rev.*, 2010, **40**(6): 649–662.

[16] Li J, Chen J, Xin B, et al., Solving the uncertain multi-objective multi-stage weapon target assignment problem via MOEA/D-AWA, *Proceedings of the* 2016 *IEEE Congress on Evolutionary Computation* (*CEC* 2016), Vancouver, Canada, July 24–29, 2016, 4934–4941.

[17] Liu B D, Zhao R Q, and Wang G, *Uncertain Programming with Applications*, Tsinghua University Press, Beijing, 2003.

[18] Das S, Mullick S S, and Suganthan P N, Recent advances in differential evolution — An updated survey, *Swarm Evol. Comput.*, 2016, **27**: 1–30.

[19] Xin B, Chen J, Zhang J, et al, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy, *IEEE Trans. Syst. Man Cybern. C — Appl. Rev.*, 2012, **42**(5): 744–767.

[20] Onwubolu G C and Davendra D, *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*, Springer-Verlag, Berlin, 2009.

[21] Wang L and Liu B, *Particle Swarm Optimization and Scheduling Algorithms*, Tsinghua University Press, Beijing, 2008.

[22] Chen J, Xin B, Peng Z H, et al, Optimal contraction theorem for exploration-exploitation tradeoff in search and optimization, *IEEE Trans. Syst. Man Cybern. A — Syst. Human.*, 2009, **39**(3): 680–691.