



MDDE: multitasking distributed differential evolution for privacy-preserving database fragmentation

Yong-Feng Ge¹ · Maria Orlowska² · Jinli Cao¹ · Hua Wang³ · Yanchun Zhang³

Received: 21 February 2021 / Revised: 29 September 2021 / Accepted: 7 November 2021 / Published online: 4 January 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

Database fragmentation has been used as a protection mechanism of database's privacy by allocating attributes with sensitive associations into separate data fragments. A typical relational database consists of multiple relations. Thus, fragmentation process is applied to each relation separately in a sequential manner. In other words, the existing database fragmentation approaches regard each relation fragmentation problem as an independent task. When solving a sequence of fragmentation problems, redundant computational resources are consumed when extracting the same fragmentation information and limit the performance of those algorithms. In this paper, a multitasking database fragmentation problem for privacy preservation requirements is formally defined. A multitasking distributed differential evolution algorithm is introduced, including a multitasking distributed framework enriched by two new operators. The introduced framework can help exchange generic and effective allocation information among different database fragmentation problems. A similarity-based alignment operator is proposed to adjust the fragment orders in different database fragmentation solutions. A perturbation-based mutation operator with adaptive mutation strategy selection is designed to sufficiently exchange evolutionary information in the solutions. Experimental results show that the proposed algorithm can outperform other competitors in terms of solution accuracy, convergence speed, and scalability.

Keywords Database fragmentation · Privacy preservation · Distributed differential evolution · Multitasking optimization

1 Introduction

Outsourced data storage can effectively reduce the overall cost of data storage and improve the scalability in storage volume [7,15,17]. Despite the tremendous benefits, outsourced data storage brings the new concern of data privacy [2,22,26,27]. Although database encryption [16,24] can protect data privacy, the additional encryption and decryption procedures reduce the query efficiency [23]. By partitioning sensitive associations and allocating the partitioned components into separate data fragments (i.e., database fragmentation), data privacy can be protected without incurring

time-consuming data transformation. Thus, the concept of vertical relational database fragmentation was introduced mainly to serve two useful purposes: efficient query processing and independently as a mechanism for data privacy protection [1,3,4].

To achieve effective privacy preservation, several data fragmentation algorithms have been proposed. In [3], encrypted vertical database fragmentation was presented to break the sensitive associations between attributes. Authors in [4] proved that the privacy-preserving database fragmentation problem belongs to the class of NP-hard problem. The authors also introduced two heuristic strategies to achieve the optimal number of fragments. In [5], the authors investigated the effectiveness of loose association in database fragmentation. In [13], a distributed memetic algorithm was proposed to tackle the trade-off between privacy and utility. In [12], an indicator named anonymity degree was introduced and used to measure the privacy preservation degree in the given database fragmentation. Although the proposed anonymity degree in [12] can effectively measure the privacy preservation degree in the database fragmentation, its performance

✉ Yong-Feng Ge
yfge93@gmail.com

¹ Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia

² Faculty of Information Technology, Polish-Japanese Academy of Information Technology, Warsaw, Poland

³ Institute for Sustainable Industries and Liveable Cities, Victoria University, Footscray, Australia

is still limited by the population diversity and information exchange efficiency. While optimizing the privacy preservation degree, two performance factors, i.e., accuracy and efficiency, are crucial. Within the same number of privacy preservation degree evaluations, an algorithm of higher accuracy can generate solutions of higher privacy preservation degrees. During the optimization, an algorithm of higher efficiency can generate solutions of the same privacy preservation degrees using fewer privacy preservation degree evaluations. Such privacy preservation evaluations cost most of the computational resources during the optimization. Therefore, we can say that the cost of the additional evaluations by the algorithms of lower efficiency is computational redundancy. In the previous algorithms, each database fragmentation problem is tackled as an independent optimization task. When optimizing a relational database containing multiple relations, effective fragmentation information extracted in one relation cannot be utilized by the others. Redundant computational resources (i.e., additional privacy preservation degree evaluations) are consumed when extracting the same fragmentation information in different fragmentation problems, which limit the optimization accuracy and efficiency of these algorithms. To improve the solution accuracy and optimization efficiency, it is worthy of developing a multitasking algorithm, which can transfer effective fragmentation information among different database fragmentation problems.

For NP-hard optimization problems, various differential evolution (DE) algorithms have been proposed [10,11,20,21]. They have shown the advantages in efficiency and reliability in optimization problems such as seismic inversion [8], microwave circuit design [28], and protein structure prediction [29]. The concept of evolutionary multitasking was first formalized in [14]. To tackle this problem, a cross-domain optimization platform was designed. Afterward, authors in [18,19] proposed two frameworks to transfer information and knowledge among different tasks. The above evolutionary multitasking algorithms only focus on continuous optimization problems. In [6], an evolutionary multitasking algorithm is designed for the capacitated vehicle routing problem. These algorithms are designed in a sequential manner, which limits the algorithms' scalability and timeliness. It is natural to think of designing a multitasking distributed framework that all the optimization tasks are solved in parallel.

In this paper, a multitasking distributed differential evolution (MDDE) algorithm is proposed to address the privacy-preserving database fragmentation problem. Each individual in MDDE represents one database fragmentation solution, and the update of the MDDE population helps identify better database fragmentation solutions. In MDDE, a multitasking distributed framework (MDF) including merge and split operators is designed, in which each database fragmentation task is optimized independently, and the effective fragmenta-

tion information is extracted and exchanged among different tasks. Moreover, a similarity-based alignment (SBA) operator is designed to adjust the fragment orders in different solutions. A perturbation-based mutation (PBM) operator with adaptive mutation strategy selection is designed to sufficiently exchange evolutionary information in the individuals. The experimental results demonstrate that the proposed MDDE can outperform the state-of-the-art algorithms in terms of solution accuracy, convergence speed, and computation efficiency.

The remainder of this paper is organized as follows. First, the multitasking database fragmentation problem is defined, and its complexity is analyzed. We then introduce the related work of the evolutionary multitasking algorithm and existing database fragmentation algorithms for privacy preservation. Afterward, we illustrate the proposed MDDE algorithm in detail. Subsequently, extensive experiments are carried out, and the experimental results are analyzed. Finally, we conclude this paper.

2 Problem definition

In this problem, we only consider vertical database fragmentation. For the simplicity of presentation, we restrict the content of the relational database to a single relation r over the relational schema R , $r(R)$. Let A be a set of attributes of R and $\underline{A} \subset A$ be the primary key of r . A fragmentation of relation r is a family of relations r_i such that the natural join of them (on \underline{A}) reconstructs the relation r . We then can define the legal fragmentation and the optimal fragmentation.

Definition 1 (*legal fragmentation*) Formally, for a given $r(R)$, we say that a fragmentation $\mathcal{F} = \{F_1, \dots, F_{NF}\}$ (i.e., a solution) is legal if:

1. $\forall i A_i \in R : \exists j F_j \in \mathcal{F}$ such that $A_i \in F_j$ (a fragmentation covers all attributes in R);
2. $\forall F_i, F_j \in \mathcal{F}, i \neq j : F_i \cap F_j = \{\underline{A}\}$ (the intersection of different fragments in the fragmentation is \underline{A});
3. $T(\mathcal{F}) \geq T_{AD}$. (The lowest anonymity degree in the fragmentation can reach the threshold.)

T_{AD} is the threshold of anonymity degree; $T(\mathcal{F})$ is the lowest anonymity degree in \mathcal{F} and calculated as:

$$T(\mathcal{F}) = \min_{1 \leq p \leq NF} (AD(F_p)) \quad (1)$$

$$AD(F_p) = \frac{NR(F_p)}{NDR(F_p)} \quad (2)$$

where $AD(F_p)$ is the anonymity degree of the p th fragment in the given fragmentation \mathcal{F} ; NF indicates the number of fragments; $NR(F_p)$ represents the number of records in

fragment F_p ; and $NNDR(F_p)$ represents the number of non-duplicate records in fragment F_p in terms of all the non-prime attributes, i.e., the number of different tuples in terms of all the non-prime attributes. Therefore, for each fragment F_p , the value of $NR(F_p)$ is higher than or equal to the value of $NNDR(F_p)$, and thus $AD(F_p) \geq 1$.

Definition 2 (*optimal fragmentation*) For $r(R)$, we say a fragmentation \mathcal{F} is optimal if:

1. $\forall \mathcal{F}'$ is legal, $AD(\mathcal{F}') \leq AD(\mathcal{F})$.

Such definition indicates the optimal fragmentation \mathcal{F} is legal and of the highest anonymity degree in all the legal fragmentations. The anonymity degree $AD(\mathcal{F})$ is calculated as:

$$AD(\mathcal{F}) = \frac{\sum_{p=1}^{NF} AD(F_p)}{NF} \tag{3}$$

In a multitasking database fragmentation problem, multiple tasks of different characteristics (e.g., data source, fragment number) are optimized in parallel. In each task, the optimization object is to identify an optimal fragmentation \mathcal{F} that can achieve the optimal anonymity degree. The optimization objective of the multitasking database fragmentation problem is to simultaneously find a set of fragmentations $\mathcal{SF} = \{\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_{NT}\}$, where NT is the number of tasks and each fragmentation in the set represents the optimal fragmentation of the corresponding task. Table 1 lists the notations in this section and the corresponding meanings.

3 Problem analysis

The analysis of the defined database fragmentation problem is a reduction from the NP-hard problem of exact cover [9], which can be formulated as follows: Given a collection \mathcal{S} of subsets of a set X , an exact cover is a subcollection \mathcal{S}' of \mathcal{S} such that each element in X is contained in exactly one subset in \mathcal{S}' . Formally, given a collection \mathcal{S} of subsets of X , an exact cover of X is a subcollection \mathcal{S}' of \mathcal{S} that:

1. The intersection of any two distinct subsets in \mathcal{S}' is empty;
2. The union of the subsets in \mathcal{S}' is X .

In the database fragmentation problem, let $X = A - \{\underline{A}\}$, where set X contains all the non-prime attributes in A . We can observe that each combination of attributes in X and \underline{A} forms a fragment F_i . With respect to the third condition in Definition 1, $\mathcal{S} = \{\forall F_i, AD(F_i) \geq T_{AD}\}$. During the construction of the database fragmentation problem, a subcollection \mathcal{S}' of \mathcal{S} is chosen. With respect to the first and second conditions in Definition 1, every attribute in X is covered by \mathcal{S}' , and every

Table 1 Notations and meanings

Notation	Meaning
r	Relation
R	Relational schema
A	Set of attributes
\underline{A}	Primary key
\mathcal{F}	Fragmentation
NF	Number of fragments
A_i	The i th attribute
F_j	The j th fragment
$T(\mathcal{F})$	The lowest anonymity degree in the fragmentation
$AD(F_p)$	Anonymity degree of p th fragment
T_{AD}	Threshold of anonymity degree
$NR(F_p)$	Number of records in the p th fragment
$NNDR(F_p)$	Number of non-duplicate records in the p th fragment
$AD(\mathcal{F})$	Anonymity degree of the fragmentation
\mathcal{SF}	Set of fragmentations
NA	Number of attributes
NR	Number of records
NT	Number of tasks

attribute appears in exact one subset (i.e., fragment) in \mathcal{S}' . As a consequence, any algorithm finding a legal fragmentation in the database fragmentation problem can be exploited to solve the exact cover problem.

4 Related work

In this section, a brief literature review of the evolutionary multitasking algorithm is first presented. Then, the related work of vertical database fragmentation for privacy preservation is introduced.

4.1 Evolutionary multitasking algorithm

The concept of evolutionary multitasking was first introduced in [14]. In this work, the concept of evolutionary multitasking was formalized. Moreover, the differences between multitasking optimization and multiobjective optimization are illustrated. A cross-domain optimization platform was developed. Authors in [18] proposed a general framework named the evolution of biocoenosis through symbiosis, which is effective in the selection of candidates from offspring and the adaptive information control among tasks. In [19], to address the multitasking optimization problem, a novel framework named symbiosis in biocoenosis optimization is proposed. The proposed framework is designed to transfer information and knowledge among different tasks. In [6], considering that the autoencoding-based explicit evolutionary multitask-

ing can only work on continuous optimization problems, the authors conducted a study toward explicit evolutionary multitasking for a combinational optimization problem, i.e., capacitated vehicle routing problem.

4.2 Database fragmentation for privacy preservation

Authors in [1] first proposed a vertical database fragmentation algorithm for data privacy preservation. In this work, all the database attributes are divided into two groups, which limits the performance in complex problems. In [3], an encrypted database fragmentation algorithm was introduced, and the advantage in protecting the sensitive association between attributes is verified. The database fragmentation problem was proved to be NP-hard in [4]. Moreover, two heuristic strategies were proposed to achieve the optimal number of fragments. A graph-based algorithm based on confidentiality constraints was proposed in [25]. The nodes in the graph were utilized to represent the database fragmentation solutions. In [5], the effectiveness of loose association in database fragmentation was investigated. The evolutionary algorithm was also utilized in database fragmentation. In [13], a distributed memetic algorithm was proposed to tackle the outsourced database fragmentation problem, in which the database privacy is satisfied, and the database utility is optimized. In [12], the anonymity degree measurement was utilized, and the anonymity degree was set as the optimization objective in database fragmentation.

5 Distributed differential evolution (DDE)

In this section, the process of the traditional DDE is illustrated. At the beginning of the evolution, the population is divided into multiple subpopulations, and each subpopulation is assigned to an island. As shown in Fig. 1, the islands are connected according to a ring topology, and the subpopulation in each island evolves independently. The chosen individual in each subpopulation is migrated to its neighbor island in the topology with a given interval. When an island receives a migrated individual, the migrated individual is inserted into the current subpopulation to replace an individual. With the help of migration, the evolutionary information in each subpopulation is exchanged. In the subpopulation, the DE algorithm is performed on individuals to improve their fitness values. In the DE algorithm, three operators, i.e., mutation, crossover, and selection, are included and described as follows.

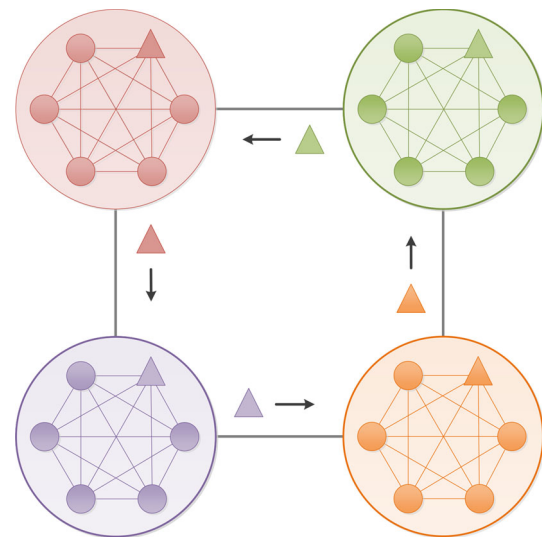


Fig. 1 An example of the migration operator in DDE, in which four subpopulations send their elite individuals (indicated by the triangles) to the neighbor subpopulations

5.1 Mutation operator

In the mutation operator, the differences between individuals are extracted and utilized to construct the mutant individuals. Various strategies have been proposed to calculate the mutant individual and listed as follows:

1. DE/rand/1

$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g) \tag{4}$$

2. DE/current-to-best/1

$$v_i^g = x_i^g + F \cdot (x_{best}^g - x_i^g) + F \cdot (x_{r_1}^g - x_{r_2}^g) \tag{5}$$

3. DE/best/1

$$v_i^g = x_{best}^g + F \cdot (x_{r_1}^g - x_{r_2}^g) \tag{6}$$

4. DE/best/2

$$v_i^g = x_{best}^g + F \cdot (x_{r_1}^g - x_{r_2}^g) + F \cdot (x_{r_3}^g - x_{r_4}^g) \tag{7}$$

5. DE/rand/2

$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g) + F \cdot (x_{r_4}^g - x_{r_5}^g) \tag{8}$$

where v_i^g indicates the i th mutant individual at generation g ; x_{best}^g represents the best individual found at generation g ; F represents the differential factor; and $r_1, r_2, r_3, r_4,$ and r_5 are indexes of the random individuals.

5.2 Crossover operator

In the crossover operator, we exchange the evolutionary information in the mutant individual \mathbf{v}_i^g and the current individual \mathbf{x}_i^g . Then, a trial individual \mathbf{u}_i^g is generated as follows:

$$\mathbf{u}_{i,j}^g = \begin{cases} \mathbf{v}_{i,j}^g, & \text{if } \text{rand}(0, 1) \leq Cr \text{ or } j = j_{\text{rand}} \\ \mathbf{x}_{i,j}^g, & \text{otherwise} \end{cases} \quad (9)$$

where $\text{rand}(0, 1)$ indicates a random float; Cr is the crossover rate; and j_{rand} is a random bit in the mutant individual to guarantee at least one bit of the trial individual comes from the mutant individual.

5.3 Selection operator

In the selection operator, the fitness values of the current individual and the generated trial individual are evaluated and then compared. The individual of higher fitness value is kept. The selection operator of a minimization problem $f(x)$ is formulated as:

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g, & \text{if } f(\mathbf{u}_i^g) \leq f(\mathbf{x}_i^g) \\ \mathbf{x}_i^g, & \text{otherwise} \end{cases} \quad (10)$$

where \mathbf{x}_i^{g+1} is the selected individual.

When the DE algorithm is utilized to solve the database fragmentation problem, each database fragmentation solution is represented by an individual in DE. The individual's fitness value indicates the anonymity degree of the database fragmentation solution. With the evolution of the population, individuals of higher fitness values are generated. Thus, we can acquire database fragmentation solutions of higher anonymity degrees.

6 Multitasking distributed differential evolution (MDDE)

In this section, an algorithm named MDDE is proposed for the multitasking database fragmentation problem. In the beginning, the representation manner of MDDE is introduced. Afterward, a framework including two operators, i.e., merge and split, is designed to migrate generic allocation information among different subproblems. Two operators named SBA and PBM are designed to sufficiently exchange evolutionary information between individuals.

6.1 Representation

In Fig. 2, an example of the representation of the MDDE algorithm is shown. As shown in the figure, a database containing nine attributes and six records is given. After the fragmentation, the given database is divided into three fragments. The first fragment contains three attributes, i.e., Area, YoB (year of birth), and ZIP code. In the second fragment, three attributes, i.e., Gender, Race, and LoS (length of stay), are allocated. Finally, attributes Type and Disease are allocated to the third fragment. Therefore, these three fragments make up a fragmentation solution shown at the bottom of the figure.

In MDDE, each individual in the optimizer represents a database fragmentation solution. To be specific, each bit of the individual indicates an attribute, and its value indicates the index of the fragment.

6.2 Multitasking distributed framework (MDF)

The MDF is proposed based on DDE. In MDF, same as DDE, the entire population is divided into multiple subpopulations, and each subpopulation evolves independently. With a given interval, the best individuals in the subpopulations are migrated among the subpopulations. There are two major differences between MDF and DDE.

- (1) Each subpopulation in DDE evolves for the same optimization problem. In MDF, each subpopulation evolves for a different optimization problem.
- (2) In DDE, the individuals in all the subpopulations represent the same optimization problem. Differently, individuals in MDF represent different optimization problems, which affects the individuals' universality.

To improve the individuals' universality in different subpopulations, two operators, i.e., merge and split, are designed.

6.2.1 Merge operator

In the migration operator, when a subpopulation receives a migrated individual, the migrated individual's evolutionary information is extracted to promote the subpopulation's search process. In the multitasking database fragmentation problem, different database fragmentation problems may contain different numbers of fragments. The merge operator is designed to decrease the number of fragments in the migrated individual.

When the number of fragments in the migrated individual exceeds the number of fragments in the subpopulation, the merge operator is applied. During the merge operator, two fragments in the individual are first chosen. Then, all the attributes in the chosen fragments are extracted and inserted into the same fragment. After the merge operator, the number

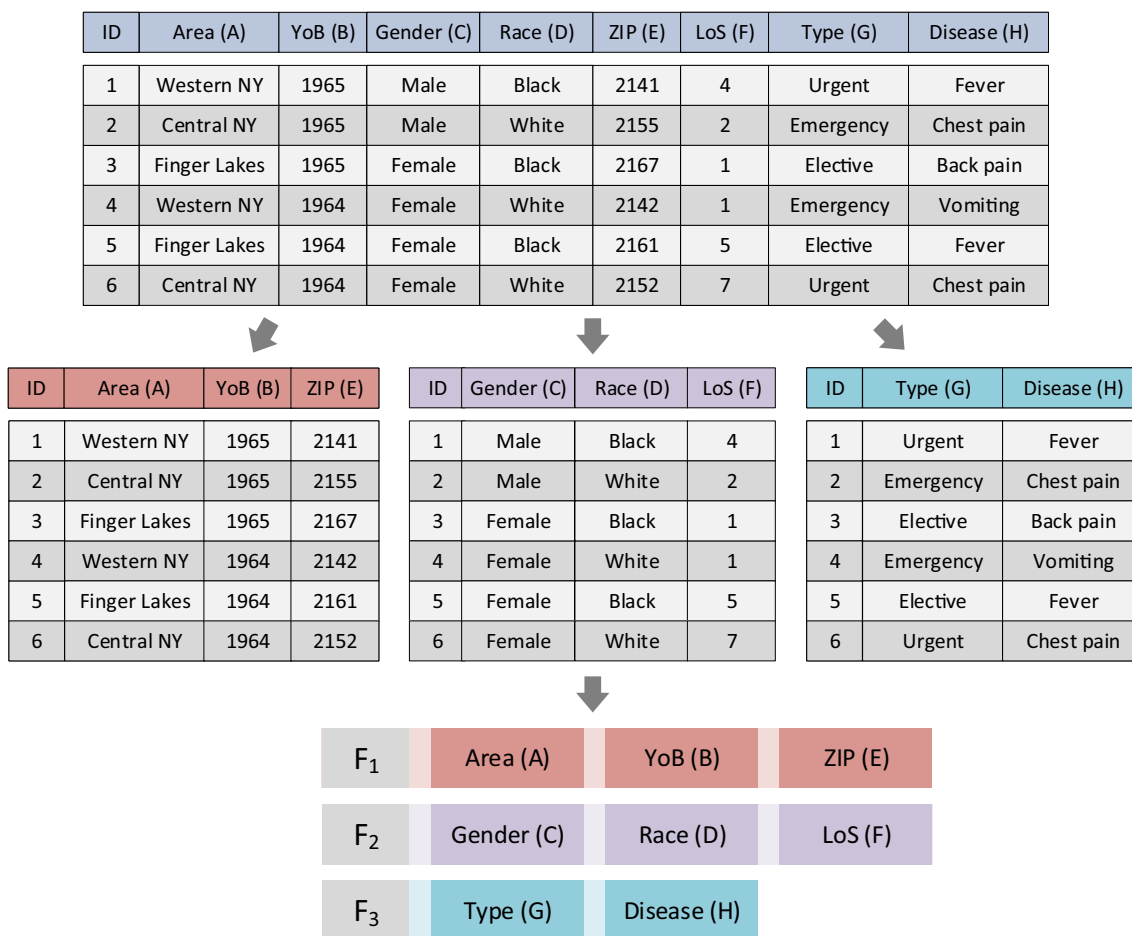


Fig. 2 In this figure, a sample database is provided, which contains nine attributes and six records. This database is separated into three fragments. The first fragment (F_1) contains attributes Area, YoB, and

ZIP. The second fragment (F_2) contains attributes Gender, Race, and LoS. Finally, the third fragment (F_3) contains attributes Type and Disease

of fragments in the migrated individual decreases. If the number of fragments in the subpopulation is still lower than the number of fragments in the migrated individual, the merge operator is applied one more time. The merge operator is executed until the numbers of fragments in the migrated individual and the subpopulation are equal.

An example of the merge operator is given in Fig. 3. As shown in the example, the migrated individual contains eight attributes, and these attributes are divided into four fragments. To be specific, the first fragment contains attributes A and F ; the second fragment contains attributes C and D . When the given individual is migrated to a subpopulation containing three fragments, the merge operator is applied. Two fragments in the given individual are randomly selected. In this example, the second fragment and the third fragment in the individual are randomly chosen and merged. Thus, the second fragment in the merged individual contains four attributes, i.e., C , D , E , and G . The number of fragments in the migrated individual is decreased to three.

With the merge operator’s help, the number of fragments in the migrated individual equals the number of fragments in the subpopulation. During the merge operator, the allocation information in the merge fragments is maintained. Also, the migrated individual can be directly inserted into the subpopulation. During the subsequent process, the migrated individual’s evolutionary information is extracted and utilized to generate the offspring.

6.2.2 Split operator

The split operator is on the other side of the merge operator. It is designed to increase the number of fragments in the migrated individual. During the split operator, one fragment in the migrated individual is chosen by random. Then, all the attributes in the chosen fragment are randomly assigned to two empty fragments. Finally, two split fragments are inserted into the fragments. After the split operator, the number of fragments in the migrated individual increases. Same

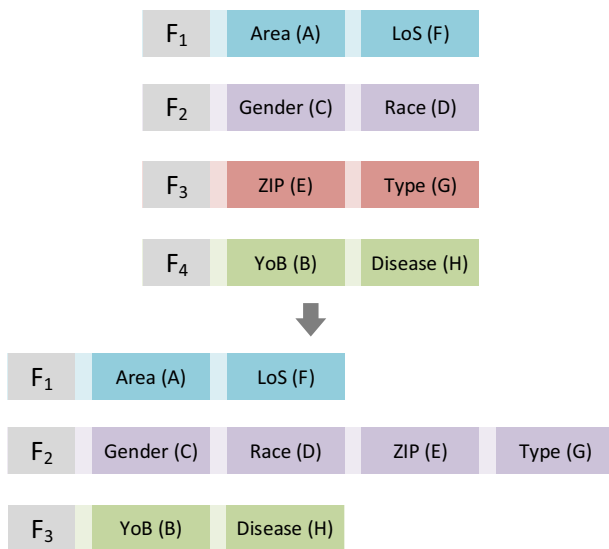


Fig. 3 An example of the merge operator, in which two fragments in the given individual (F_2 and F_3) are merged into one fragment. A new individual containing the merged fragment (F_2) is generated

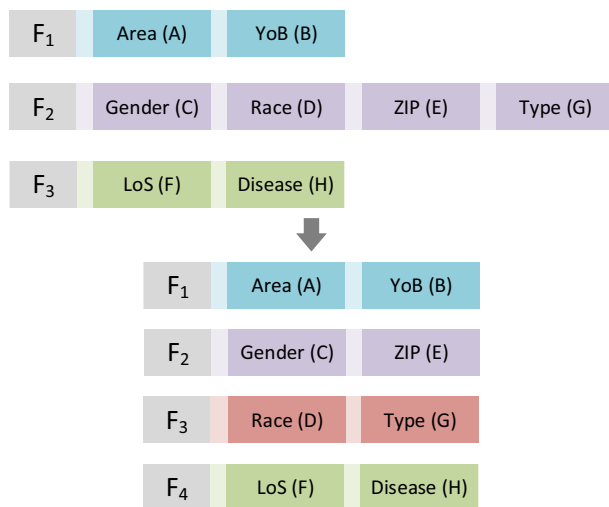


Fig. 4 An example of the split operator, in which the second fragment (F_2) in the given individual is split into two fragments. A new individual containing the split two fragments (F_2 and F_3) is generated

as the merge operator, the split operator is executed until the numbers of fragments in the migrated individual and the subpopulation are equal.

As shown in Fig. 4, an example of the split operator is given. In the example, the migrated individual contains eight attributes and three fragments. The first fragment contains attributes *A* and *B*, while the second fragment involves attributes *C*, *D*, *E*, and *G*. During the split operator, the second fragment is chosen randomly. Attributes *C*, *E* are assigned to one fragment, and attributes *D*, *G* are assigned to the other fragment. After the split operator, the number of

fragments in the migrated individual increases from three to four.

The split operator helps guarantee that the migrated individual and the subpopulation are of the same fragment number. The allocation information in the split fragment is partially maintained in the new fragments. During the afterward evolution, the migrated individual’s allocation information is exchanged with the other individuals in the subpopulation, and offspring is generated.

6.3 Similarity-based alignment (SBA)

When updating the database fragmentation solutions, the DE algorithm is performed. In DE, mutation and crossover operators are utilized for exchanging allocation information (i.e., which attribute is allocated to which fragment) among different solutions. In the database fragmentation problem, different fragments are of the same property, and their order does not affect the result. If two database fragmentation solutions assign the same attributes to the fragments and their fragment orders are different, these two solutions are still regarded as the same solution. When exchanging allocation information between individuals, their fragment orders matter. If the fragment orders of two database fragmentation solutions are similar, the difference in their allocation is easy to extract. Otherwise, the difference between two database fragmentation solutions is challenging to identify. In this algorithm, the SBA operator is designed to adjust all the database fragmentation solutions’ fragment orders.

In the SBA, the best individual of each subpopulation is chosen as the alignment standard. Then, all the other individuals are adjusted according to the given alignment standard. First, for a given individual, all its fragments are compared with the first fragment in the alignment standard, and the similarity degrees are calculated. The fragment in the individual with the highest similarity is extracted and reallocated to the first position. Then, all the individuals’ left fragments are compared with the second fragment in the alignment standard, and the similarity degrees are calculated. The fragment with the highest similarity is reallocated to the second position. The individual is finally reallocated according to the alignment standard.

An example of the SBA is given in Fig. 5. The best individual in the subpopulation is given and set as the alignment standard. All the given individual fragments are compared with the first fragment of the alignment standard. In this example, the third fragment in the given individual is of the highest similarity. Then, the third fragment is reallocated to the first position. Afterward, the other fragments are compared with the second fragment in the alignment standard, and the second fragment is chosen. The second fragment in the given individual is reallocated to the second position. Similarly, the left fragment in the best individual (i.e., the

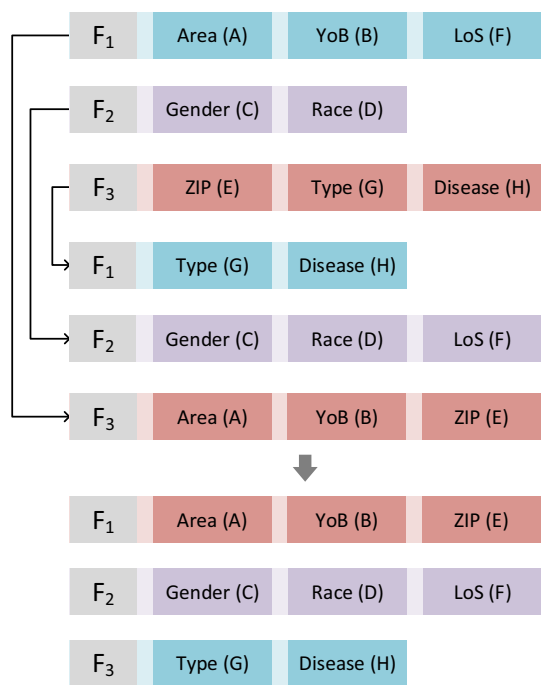


Fig. 5 An example of the SBA operator, in which the fragment order in the given individual is adjusted according to the best individual. Accordingly, a new individual is generated

third fragment) matches the first fragment in the given individual. The orders of fragments in the given individual are adjusted accordingly.

After performing the SBA operator, all the individuals' fragment orders are adjusted according to the given alignment standard. In these individuals, all the fragments of the same position contain similar attributes. The differences in fragment allocation between two individuals can be directly identified by comparing the fragments at the same position in these individuals. Thus, during the afterward mutation and crossover operators, the same allocation information in the individuals can be kept, and the differences between individuals can be extracted and exchanged.

6.4 Perturbation-based mutation (PBM)

In the traditional mutation operator, two parts are involved. The first part is a chosen individual acting as the base. The second part is the calculation result of the chosen individuals. We can also regard the second part as a perturbation for the base part. Taking "DE/rand/1" strategy as an example:

$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g) \tag{11}$$

we can see the first part (i.e., base part) is the first random individual, and the second part (i.e., perturbation part) is acquired by calculating the difference between the second and third

random individuals. When tackling the continuous optimization problem, it is natural to directly calculate the mutant individual according to the given calculation rule. However, when tackling the discrete database fragmentation problem, these calculation rules cannot maintain their effectiveness.

Based on the "base-perturbation" manner, two perturbation-based mutation strategies are proposed. The first mutation strategy is a variant of the traditional "DE/rand/1" strategy, which can help improve population diversity and exploration ability. In this variant, three individuals are chosen randomly. The first random individual acts as the base. Then, the difference between the latter two individuals is extracted as the perturbation. Finally, the perturbation information is added to the base part, and the mutation individual is generated. The second mutation strategy is a variant of the traditional "DE/best/1" strategy, which helps maintain the exploitation ability. In this variant, the best individual and two random individuals are utilized. The best individual serves as the base, while the difference between two random individuals serves as the perturbation.

An example of two "base-perturbation" mutation strategies is given in Fig. 6. In the random mutation strategy, three random individuals are utilized. The difference between the second individual x_{r_2} and the third random individual x_{r_3} is extracted. To be specific, on the first, fourth, fifth, and seventh bits, the attributes are assigned to the same fragment. Thus, no difference is extracted, and the corresponding values are directly copied to the perturbation vector p . On the other attributes, they are assigned to different fragments in two individuals. Thus, for each attribute, one fragment is randomly selected from two individuals. On the second and eighth bits, attributes are assigned according to x_{r_2} , while attributes on the third and sixth bits are assigned according to x_{r_3} . Afterward, the difference between two random individuals is extracted and used to construct the perturbation vector. Finally, the generated perturbation vector p is combined with the base individual, and the mutant individual is generated. On the first, fourth, fifth, and eighth bits, these two vectors contain the same allocation information and are kept in the mutant individual. On the third and sixth bits, the mutant individual's allocation information comes from the base individual. On all the other bits, the allocation information comes from the perturbation vector.

In the best mutation strategy, the same rule is applied. The best individual in the subpopulation acts as the base. Two random individuals are used to generate the perturbation vector. Then, the mutant individual is generated based on the base part and the perturbation vector. On the second, third, seventh, and eighth bits, the contents from the base part and the perturbation vector are the same and kept in the mutant individual. On the fourth and sixth bits, the allocation information comes from the base individual. On the left bits, the allocation information comes from the perturbation vector.

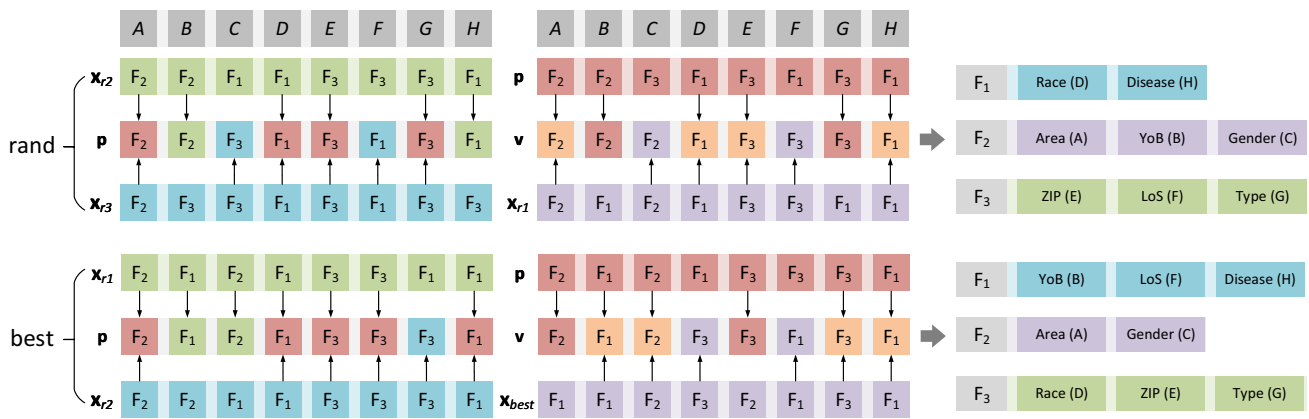


Fig. 6 An example of the PBM operator, in which the random mutation strategy is utilized on individuals x_{r1} , x_{r2} , and x_{r3} , while the best mutation strategy is utilized on individuals x_{best} , x_{r1} , and x_{r2}

In the “base-perturbation” mutation strategies, the individuals’ evolutionary information is divided into two parts. The first one is the base part. In the best mutation strategy, the base part is the best individual in the subpopulation, which can help improve the subpopulation’s exploitation ability. In the random mutation strategy, the base part is a random individual in the subpopulation, which helps maintain the population diversity and exploration ability. The second one is the perturbation part, which extracts the difference between random individuals. Finally, the allocation information from the base individual and the perturbation vector is combined. The same allocation information is kept, while the different allocation information is randomly selected.

6.5 Adaptive mutation strategy selection

In the above section, two mutation strategies are proposed, i.e., random mutation strategy and best mutation strategy. In the random mutation strategy, a random individual is set as the base, which helps improve the exploration ability. On the other hand, in the best mutation strategy, the best individual in the subpopulation is set as the base, which helps improve the exploitation ability. During the evolution, it is crucial to achieving the trade-off between exploration ability and exploitation ability. To balance the exploration ability and exploitation ability, the mutation strategy is adaptively selected.

At the beginning of the evolution, for each individual, a mutation strategy is chosen by random. Then, in the first generation, the chosen mutation strategy is performed. During the afterward selection operator, if the generated trial individual is better than the current individual, the chosen mutation strategy remains unchanged. If the generated trial individual is not better, the chosen mutation strategy is changed to the other one. If the current mutation strategy is the random

mutation strategy, it will be changed to the best mutation strategy and vice versa.

The adaptive mutation strategy selection can help achieve the trade-off between the exploration ability and the exploitation ability. If the chosen mutation strategy can help generate better offspring during the evolution, which means the chosen mutation strategy is effective for the evolution, it will be kept. If the chosen mutation strategy does not help generate more competitive offspring, it will be replaced. In different evolutionary stages, exploration ability and exploitation ability show different effectiveness. Different mutation strategies are needed for different individuals, and the above adaptive selection strategy can provide them different mutation strategies according to the performance.

6.6 Crossover operator

The crossover operator is designed to combine the allocation information of the mutant individual and the current individual to generate the trial individual. With the predefined crossover rate C_r , the allocation information is extracted from the mutant individual. With the $1 - C_r$ possibility, the allocation information comes from the current individual. Furthermore, one bit j_{rand} is randomly chosen to guarantee that at least one bit of the trial individual comes from the mutant individual.

Figure 7 shows an example of the crossover operator. The mutant individual and the current individual are given. The sixth bit is chosen as the j_{rand} bit, and the allocation information on this bit comes from the mutant individual. Moreover, with the crossover rate C_r , the allocation information on the second, fourth, and seventh bits is also extracted from the mutant individual. On all the other bits, the trial individual’s attributes are assigned according to the current individual.

After the crossover operator, the allocation information in the mutant individual and the current individual is sufficiently

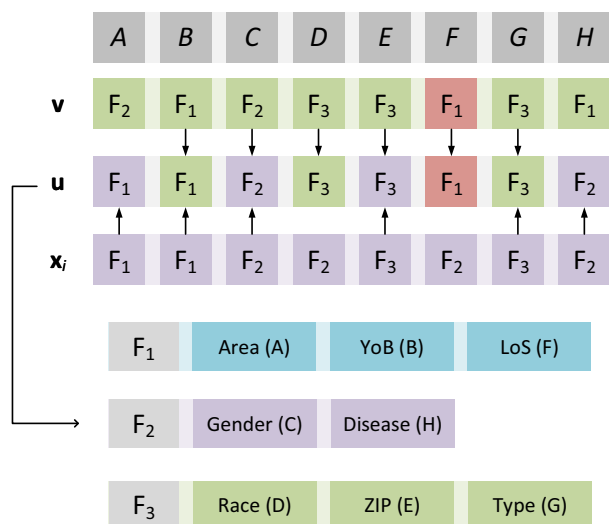


Fig. 7 An example of the crossover operator, in which the mutant individual v and the current individual x_i exchange the allocation information and generate the trial individual u

Algorithm 1 Pseudocode of MDDEspsmasternodesps

```

1: procedure MASTER NODE
2:   Assign the database fragmentation problems to slave nodes
3:   Set initial generation  $g = 0$ 
4:   Divide the population into  $N$  subpopulations
5:   while terminal condition is not met do
6:     if  $g \% MI = 0$  then
7:       Receive migrated individuals from slave nodes
8:       Send the migrated individuals to the corresponding nodes
9:     end if
10:     $g = g + 1$ 
11:  end while
12:  Collect the best solutions for the given database fragmentation problems
13:  Output the best solutions
14: end procedure
    
```

exchanged, and a trial individual is generated. In the trial individual, the percentage of allocation information source is decided by the predefined crossover rate Cr .

6.7 Overall process

In this section, the pseudo-code of the proposed MDDE algorithm is given in Algorithms 1 and 2. As shown in the pseudo-code, the proposed algorithm is implemented in a master–slave manner. At the master node, the corresponding database fragmentation problems are assigned to the corresponding slave nodes. Then, the generation counter is initialized as zero. The entire population is divided into N subpopulations, and each subpopulation is assigned to one slave node. With the migration interval MI , the master node receives the migrated individuals from the slave nodes and sends the migrated individuals to the corresponding slave nodes. In the master node, the migration operator is executed

Algorithm 2 Pseudocode of MDDEspsslavenodesps

```

1: procedure SLAVE NODE
2:   for each individual do
3:     Randomly choose a mutation strategy from the pool
4:   end for
5:   for every generation do
6:     Perform the SBA operator
7:     for each individual do
8:       Perform the PBM operator
9:       Perform the crossover operator
10:      Perform the selection operator
11:      if the generated individual is better then
12:        Replace the current individual by the trial individual
13:      else
14:        Change the current mutation strategy
15:      end if
16:    end for
17:    if  $g \% MI = 0$  then
18:      Send the best individual to the master node
19:      Receive the migrated individual from the master node
20:      if Migrated individual is with higher fragment number
21:        Perform merge operator
22:      end if
23:      if Migrated individual is with lower fragment number then
24:        Perform split operator
25:      end if
26:      Replace a randomly chosen individual
27:    end if
28:  end for
29: end procedure
    
```

until the termination condition is met. The best solutions for all the database fragmentation problems from all the slave nodes are collected by the master node and outputted.

In each slave node, one database fragmentation problem and one subpopulation are assigned at the beginning of the evolution. For each individual in the subpopulation, one mutation strategy is selected by random. In every generation, the SBA operator is performed to adjust the fragment order. The PBM operator and crossover operator are performed on all the individuals, and trial individuals are generated. If the trial individual is better than the corresponding current individual, the current individual is replaced by the trial individual. Otherwise, the current individual is kept, and the mutation strategy is changed. During the evolution, with the migration interval MI , each subpopulation sends the best individual to the master node and receives a migrated individual from the master node. If the migrated individual contains a higher fragment number, the proposed merge operator is executed. If the migrated individual contains a lower fragment number, the proposed split operator is executed. Then, the migrated individual contains the same fragment number, and one randomly chosen individual in the subpopulation is replaced.

Based on the problem definition, the time complexity of calculating the anonymity degree for a given fragmentation

is $\mathcal{O}(\text{NA} \times \text{NR} \times \log \text{NR})$. According to the overall process of the proposed MDDE, its time complexity is $\mathcal{O}(\text{NA} \times \text{NR}^2 \times \log \text{NR})$ and space complexity is $\mathcal{O}(\text{NA} \times \text{NR} \times \text{NT})$.

7 Experimental result

In this section, the results of various experiments are presented. First, the experimental setup and performance metric are introduced. Then, the proposed MDDE algorithm is compared with competitive database fragmentation algorithms and state-of-the-art evolutionary multitasking algorithms in terms of accuracy and efficiency. Afterward, the impact of the proposed components in MDDE is investigated, and the speedup ratio of MDDE is calculated. Finally, the parameter setting of MDDE is discussed.

7.1 Experimental setup

In the experiments, 16 test cases are utilized to verify the proposed MDDE algorithm's effectiveness in database fragmentation. These test cases are generated based on the public datasets of the New York State Department of Health.¹ In each test case, different characteristics (e.g., fragment number, data source) are assigned to different tasks. Table 2 outlines the properties of these test cases, which include numbers of records NR, numbers of sample records NSR, numbers of attributes NA, numbers of tasks NT, numbers of fragments NF, and numbers of data sources NDS.

In the proposed MDDE algorithm, the number of subpopulations in MDDE is set as NT; the subpopulation size SPS is set as 10; the migration interval MI is set as 10; and the crossover rate Cr is set as 0.5. For all the algorithms, the anonymity degree threshold T_{AD} is set as 2.0; the maximum fitness evaluation number maxNFEs is set as $\text{NR}/100$. Furthermore, in each test case, the proposed MDDE and the compared approaches are performed in 25 independent runs.

The island model of MDDE is implemented by the Message Passing Interface (MPI). Each subpopulation is assigned to a computation core in the CPU. The communication between subpopulations is implemented by the message exchange between CPU cores. Moreover, MDDE and all the compared algorithms in this paper are implemented in C++ and performed on a local cluster containing 60 compute nodes (OS: Ubuntu 16.04; CPU: 3.40 GHz 4-Core Intel i5-7500; Memory: 8 GB).

¹ <https://health.data.ny.gov/Health/Hospital-Inpatient-Discharges-SPARCS-De-Identified/82xm-y6g8>.

Table 2 Properties of 16 test cases

Test cases	NR	NSR	NA	NT	NF	NDS
T_1	100,000	10,000	31	4	2, 3, 4, 5	1
T_2	100,000	20,000	31	4	2, 3, 4, 5	1
T_3	100,000	30,000	31	4	2, 3, 4, 5	1
T_4	100,000	40,000	31	4	2, 3, 4, 5	1
T_5	200,000	10,000	36	4	2, 3, 4, 5	1
T_6	200,000	20,000	36	4	2, 3, 4, 5	1
T_7	200,000	30,000	36	4	2, 3, 4, 5	1
T_8	200,000	40,000	36	4	2, 3, 4, 5	1
T_9	300,000	10,000	31	4	2, 3, 4, 5	4
T_{10}	300,000	20,000	31	4	2, 3, 4, 5	4
T_{11}	300,000	30,000	31	4	2, 3, 4, 5	4
T_{12}	300,000	40,000	31	4	2, 3, 4, 5	4
T_{13}	400,000	10,000	36	4	2, 3, 4, 5	4
T_{14}	400,000	20,000	36	4	2, 3, 4, 5	4
T_{15}	400,000	30,000	36	4	2, 3, 4, 5	4
T_{16}	400,000	40,000	36	4	2, 3, 4, 5	4

7.2 Performance metric

The average anonymity degree \overline{AD} is defined as the performance metric, which calculates the average value of AD obtained by all the fragmentations in \mathcal{SF} . More formally, this metric is calculated as:

$$\overline{AD}(\mathcal{SF}) = \frac{\sum_{t=1}^{\text{NT}} \text{AD}(\mathcal{F}_t)}{\text{NT}} \quad (12)$$

where \mathcal{F}_t is the obtained fragmentation of t th task; $\text{AD}(\mathcal{F}_t)$ indicates its anonymity degree; and NT is the task number.

7.3 Comparisons with competitive database fragmentation algorithms

To verify the effectiveness of the proposed MDDE algorithm in test cases, three competitive database fragmentation approaches, i.e., DE [21], S-DDE [12], and HA [4] are compared. The specific descriptions of these three algorithms are listed as follows:

- (1) DE [21]: This approach acts as a baseline algorithm. The differences between DE and MDDE in performance show the effectiveness of the proposed operators in MDDE.
- (2) S-DDE [12]: In this algorithm, the discrete database fragmentation problem is optimized by the proposed set-based mutation and crossover operators.
- (3) HA [4]: This is a state-of-the-art heuristic algorithm for database fragmentation problems, in which two heuristic

Table 3 Comparisons with competitive database fragmentation algorithms

Test cases	DE		S-DDE		HA		MDDE	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
T_1	1.69E+02	4.92E+01	1.47E+02	5.85E+01	1.14E+02	1.18E+02	3.21E+02[†]	1.29E+02
T_2	6.11E+02	1.96E+02	7.10E+02	2.70E+02	4.07E+02	2.59E+02	7.87E+02[†]	2.45E+02
T_3	1.35E+03	3.71E+02	1.17E+03	3.69E+02	8.05E+02	4.00E+02	1.72E+03[†]	3.02E+02
T_4	2.22E+03	3.36E+02	1.90E+03	5.03E+02	1.11E+03	6.78E+02	3.45E+03[†]	5.73E+02
T_5	3.18E+01	2.30E+01	4.74E+01	3.10E+01	3.70E+01	5.14E+01	6.47E+01[†]	6.14E+01
T_6	3.16E+02	1.33E+02	2.93E+02	2.28E+02	1.96E+02	2.30E+02	4.77E+02[†]	1.38E+02
T_7	6.49E+02	2.21E+02	6.24E+02	1.57E+02	2.97E+02	2.68E+02	1.05E+03[†]	3.76E+02
T_8	1.24E+03	4.50E+02	9.74E+02	3.60E+02	4.77E+02	3.44E+02	1.59E+03[†]	5.02E+02
T_9	3.07E+02	1.11E+02	3.59E+02	1.98E+02	2.00E+02	1.61E+02	3.91E+02[†]	1.20E+02
T_{10}	8.97E+02	2.44E+02	1.08E+03	2.68E+02	7.30E+02	3.16E+02	1.23E+03[†]	2.49E+02
T_{11}	1.82E+03	4.41E+02	1.78E+03	4.00E+02	1.30E+03	5.35E+02	3.11E+03[†]	6.38E+02
T_{12}	2.96E+03	4.95E+02	2.56E+03	5.72E+02	1.72E+03	6.45E+02	4.60E+03[†]	8.19E+02
T_{13}	1.18E+02	5.24E+01	1.10E+02	4.79E+01	1.20E+02	1.13E+02	2.18E+02[†]	1.00E+02
T_{14}	5.44E+02	1.45E+02	4.42E+02	1.53E+02	3.60E+02	2.64E+02	7.88E+02[†]	2.46E+02
T_{15}	1.11E+03	3.72E+02	9.49E+02	3.50E+02	7.53E+02	4.63E+02	1.23E+03[†]	3.53E+02
T_{16}	1.81E+03	4.64E+02	1.34E+03	3.92E+02	1.16E+03	5.60E+02	2.13E+03[†]	3.79E+02

strategies are designed to achieve the optimal fragmentation. Moreover, the correctness and complexity of the proposed heuristic strategies are analyzed.

The average and standard deviation values obtained by these approaches are calculated and listed in Table 3. The best results are labeled in **boldface**. Overall, MDDE algorithm can outperform the compared algorithms in all the test cases. Compared with DE and S-DDE, the advantage of the proposed framework and operators is verified. First, MDF helps maintain population diversity and improve exploration ability. Moreover, the individuals' allocation information is effectively extracted by the SBA and PBM operators. Due to the advantage of population diversity, MDDE can achieve a better performance than HA. In HA, the search direction is led by the predefined heuristic strategy, and its search direction is narrow. In complex test cases, HA is more likely to get trapped by the local optima. Besides, Wilcoxon rank-sum (significance level 0.05) is employed to verify MDDE's advantage in a statistical sense. As shown in Table 3, symbol [†] shows the labeled results can achieve significant differences from the compared results.

The obtained convergence curves in six typical test cases are plotted in Fig. 8. Four lines out of seven indicate the convergence curves obtained by DE, S-DDE, HA, and MDDE, respectively. For each point, the value on the horizontal axis represents the number of fitness evaluations, while the vertical axis represents the average anonymity degrees obtained in 25 independent runs. In the beginning, all three algorithms converge rapidly. HA quickly gets trapped by the local

optima and stagnates. Thanks to the exploration ability of DE, S-DDE, and MDDE, they can continuously improve the anonymity degree during the search. The differences between the lines of MDDE and the lines of DE verify the effectiveness of the proposed framework and operators in MDDE. Overall, MDDE can achieve the highest convergence speed due to the trade-off between exploration and exploitation.

More specifically, in Table 4, ten values of \overline{AD} with the same margin are listed, and the corresponding values of NFEs cost by these four algorithms are calculated. The best results are labeled in **boldface**. For example, in the first line of the DE algorithm, the value 2.00E+02 indicates that the DE algorithm costs 200 fitness evaluations to achieve value 340 in anonymity degree. The label “-” indicates that the corresponding algorithm cannot achieve the given value. Moreover, the ratio of NFEs of each compared algorithm is calculated by dividing NFEs of MDDE into NFEs achieved by the compared algorithm. If the value of ratio exceeds 100%, it indicates that the compared algorithm costs more fitness evaluations to achieve the same value of \overline{AD} as MDDE. Correspondingly, the MDDE algorithm shows higher efficiency when reaching the chosen value of \overline{AD} . In contrast, it indicates that the compared algorithm costs fewer fitness evaluations to reach the same value of \overline{AD} as MDDE. Thus, the compared algorithm is more efficient when achieving the value of \overline{AD} . Comparing DE and MDDE, we can see that DE can reach six values in the list, while MDDE can achieve all ten values. In other words, DE cannot achieve the same accuracy degree as MDDE. Focusing on the ratio, we can find that the first three ratios in DE are equal to 100%, which indi-

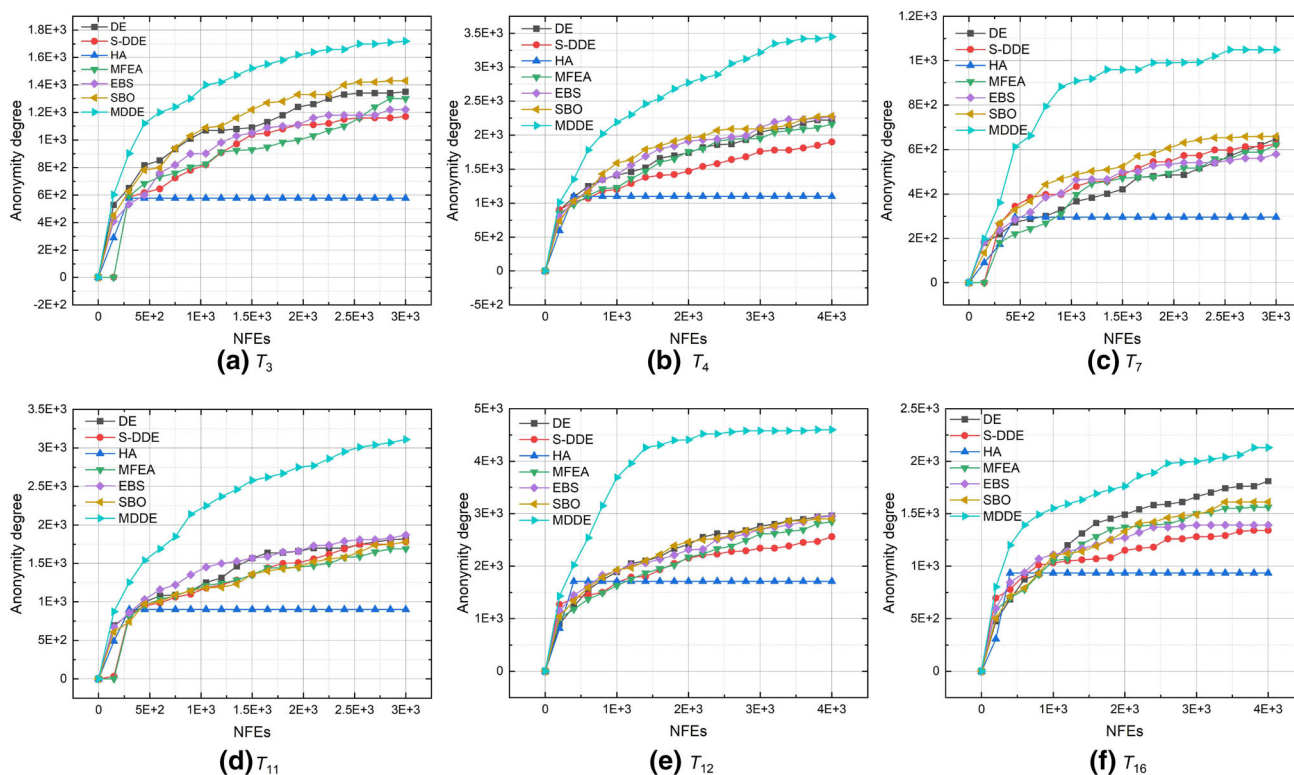


Fig. 8 Convergence curves of MDDE and all the compared algorithms in six typical test cases

icates that DE can achieve the same efficiency as MDDE at the beginning of the evolution. Afterward, limited by the population diversity and information sharing, DE cannot keep the same optimization efficiency as MDDE. Similarly, when comparing S-DDE and MDDE, we can see that the S-DDE algorithm shows lower optimization accuracy than MDDE. Five ratios of S-DDE are higher than or equal to 100%, which indicates that S-DDE cannot achieve higher efficiency than MDDE during the optimization. In HA, all the listed three ratios are equal to or higher than 100%, which indicates that HA cannot maintain the same efficiency as MDDE. As mentioned above, with the help of the predefined heuristic strategy, HA can achieve quick convergence at the beginning of the optimization. Also, led by the heuristic strategy, its search direction is narrow, and it is more likely to be trapped by the local optima. Its optimization accuracy cannot be guaranteed. Overall, the proposed MDDE can achieve the highest accuracy in the comparison. Also, its optimization efficiency is equal to or higher than the compared algorithms during the entire optimization.

Based on the original papers, the time complexity of HA is $\mathcal{O}(NA^2 \times NR \times \log NR)$. Compared with MDDE, since the value of NA is lower than NR in the test cases, the time complexity of HA is slightly lower than MDDE. On the other hand, the time complexity of the other two evolutionary algorithms (DE and S-DDE) is $\mathcal{O}(NA \times NR^2 \times \log NR)$, which

is the same as MDDE. In addition, the space complexity of DE, S-DDE, and HA is $\mathcal{O}(NA \times NR \times NT)$, which is the same as the space complexity of MDDE. To sum up, in terms of time complexity, HA shows its advantage, while the other algorithms are the same as MDDE. In terms of space complexity, MDDE and all the compared algorithms do not show significant differences. Moreover, the space consumption of MDDE and the compared algorithms is plotted in Fig. 9. In each test case, the space consumption of different algorithms does not show a significant difference, which is also reflected by the above analysis regarding the space complexity. In these test cases, most of the space is occupied by the records in the tackled databases, and the minor differences between different algorithms are caused by the different data structures utilized during the optimization.

7.4 Comparisons with state-of-the-art evolutionary multitasking algorithms

To further verify the effectiveness of the proposed MDDE algorithm, three state-of-the-art evolutionary multitasking algorithms, i.e., MFEA [14], EBS [18], and SBO [19], are compared. The specific descriptions of these three algorithms are listed as follows:

Table 4 Comparisons of MDDE and the compared algorithms in terms of accuracy and efficiency on T_4

AD	DE		S-DDE		HA		MFEA		EBS		SBO		MDDE	
	NFEs	Ratio (%)	NFEs	Ratio (%)	NFEs	Ratio (%)	NFEs	Ratio (%)	NFEs	Ratio (%)	NFEs	Ratio (%)	NFEs	Ratio (%)
3.40E+02	2.00E+02	100.00	2.00E+02	100.00	2.00E+02	100.00	2.00E+02	100.00	2.00E+02	100.00	2.00E+02	100.00	2.00E+02	100.00
6.80E+02	2.00E+02	100.00	2.00E+02	100.00	4.00E+02	200.00	2.00E+02	100.00	2.00E+02	100.00	2.00E+02	100.00	2.00E+02	100.00
1.02E+03	4.00E+02	100.00	4.00E+02	100.00	4.00E+02	100.00	6.00E+02	150.00	4.00E+02	100.00	4.00E+02	100.00	4.00E+02	100.00
1.36E+03	1.00E+03	166.67	1.40E+03	233.33	-	-	1.20E+03	200.00	1.00E+03	200.00	8.00E+02	166.67	6.00E+02	133.33
1.70E+03	1.80E+03	300.00	3.00E+03	500.00	-	-	2.00E+03	333.33	1.60E+03	266.67	1.40E+03	233.33	6.00E+02	233.33
2.04E+03	3.00E+03	300.00	-	-	-	-	3.20E+03	320.00	3.00E+03	300.00	2.40E+03	240.00	1.00E+03	240.00
2.38E+03	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2.72E+03	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3.06E+03	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3.40E+03	-	-	-	-	-	-	-	-	-	-	-	-	-	-

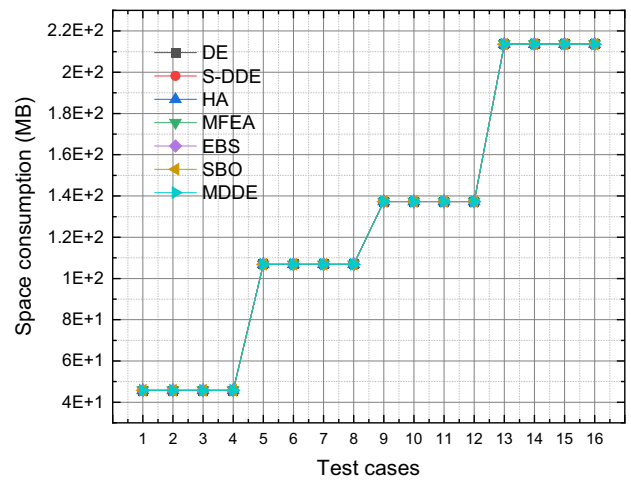


Fig. 9 Space consumption of MDDE and all the compared algorithms in all the test cases

- (1) MFEA [14]: In this algorithm, the multitasking problem is solved by a cross-domain optimization platform.
- (2) EBS [18]: In this algorithm, a novel framework is proposed to deal with selecting candidates from offspring and the adaptive control of information exchange among tasks.
- (3) SBO [19]: A framework named symbiosis in biocoenosis optimization is designed in this approach to transfer information among tasks.

The average and standard deviation values obtained by all the algorithms are listed in Table 5, and the best results are labeled in **boldface**. The proposed MDDE algorithm can outperform the compared algorithms in all the test cases. With the help of merge and split operators, individuals can be migrated among subpopulations for different tasks, and the generic allocation information in the migrated individuals can be extracted and utilized. As shown in the table, the results of the statistical analyses are represented by the symbol †. Through analyzing the comparison results, it is clear that the MDDE algorithm can obtain significantly better results than the other compared algorithms in all of the test cases.

The obtained convergence curves in six typical test cases are plotted in Fig. 8. Four lines out of seven indicate the convergence curves obtained by MFEA, EBS, SBO, and MDDE, respectively. In the beginning, all four algorithms converge rapidly. Thanks to the population diversity of MDDE, it can achieve continuous convergence. The MDDE algorithm shows its advantage in exploitation ability. The MDDE algorithm continuously achieves better results during the entire evolution. Overall, MDDE can achieve the highest convergence speed due to the trade-off between exploration and exploitation.

Table 5 Comparisons with state-of-the-art evolutionary multitasking algorithms

Test cases	MFEA		EBS		SBO		MDDE	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
T_1	1.94E+02	8.52E+01	1.74E+02	8.69E+01	1.72E+02	6.01E+01	3.21E+02[†]	1.29E+02
T_2	7.03E+02	3.73E+02	6.87E+02	2.43E+02	6.67E+02	2.34E+02	7.87E+02[†]	2.45E+02
T_3	1.30E+03	3.73E+02	1.22E+03	3.52E+02	1.43E+03	3.40E+02	1.72E+03[†]	3.02E+02
T_4	2.16E+03	4.71E+02	2.27E+03	5.18E+02	2.28E+03	5.13E+02	3.45E+03[†]	5.73E+02
T_5	3.76E+01	2.93E+01	3.61E+01	2.55E+01	3.64E+01	2.26E+01	6.47E+01[†]	6.14E+01
T_6	2.89E+02	1.42E+02	2.92E+02	1.36E+02	3.44E+02	1.22E+02	4.77E+02[†]	1.38E+02
T_7	6.22E+02	1.79E+02	5.79E+02	1.54E+02	6.59E+02	1.38E+02	1.05E+03[†]	3.76E+02
T_8	1.09E+03	3.74E+02	1.05E+03	4.55E+02	1.30E+03	4.47E+02	1.59E+03[†]	5.02E+02
T_9	3.80E+02	2.26E+02	2.33E+02	4.80E+01	2.38E+02	4.31E+01	3.91E+02[†]	1.20E+02
T_{10}	1.01E+03	2.43E+02	8.70E+02	2.35E+02	8.71E+02	2.41E+02	1.23E+03[†]	2.49E+02
T_{11}	1.69E+03	3.86E+02	1.87E+03	3.31E+02	1.78E+03	3.47E+02	3.11E+03[†]	6.38E+02
T_{12}	2.84E+03	4.43E+02	2.95E+03	5.59E+02	2.90E+03	4.37E+02	4.60E+03[†]	8.19E+02
T_{13}	9.83E+01	5.26E+01	1.27E+02	7.07E+01	1.27E+02	7.39E+01	2.18E+02[†]	1.00E+02
T_{14}	4.95E+02	1.77E+02	5.13E+02	1.67E+02	5.80E+02	2.29E+02	7.88E+02[†]	2.46E+02
T_{15}	9.57E+02	3.09E+02	9.83E+02	3.20E+02	1.08E+03	3.56E+02	1.23E+03[†]	3.53E+02
T_{16}	1.56E+03	4.73E+02	1.39E+03	3.60E+02	1.61E+03	4.79E+02	2.13E+03[†]	3.79E+02

Further, according to the data presented in Table 4, we can analyze these algorithms in terms of accuracy and efficiency. MFEA, EBS, and SBO algorithms all reach six \overline{AD} values, which is lower than the value achieved by MDDE (i.e., ten). In other words, when utilizing the same number of fitness evaluations, these compared three algorithms cannot achieve the same accuracy degree as MDDE. Moreover, when comparing the efficiency of MFEA and MDDE, we can see the first two ratios of MFEA are 100%, which indicates MFEA can achieve the same efficiency as MDDE at the beginning of the optimization. Afterward, the ratios of MFEA are all higher than 100%, which indicates that MDDE can acquire the same values of \overline{AD} by costing lower numbers of fitness evaluations. Similarly, the ratios achieved by EBS and SBO algorithms are also equal to 100% at the beginning of the optimization. Afterward, their ratios exceed 100%. Such comparisons indicate that the proposed components (i.e., MDF, SBA, and PBM) can enhance the performance of MDDE in terms of optimization accuracy and efficiency.

According to the original papers, the time complexity of all the compared multitasking evolutionary algorithm is $\mathcal{O}(NA \times NR^2 \times \log NR)$, which is the same as MDDE. Since MDDE is implemented based on the distributed framework, parallel computation can help reduce the running time and achieve speedup. In terms of space consumption, the space complexity of MFEA, EBS, and SBO is $\mathcal{O}(NA \times NR \times NT)$, which is the same as the space complexity of MDDE. To sum up, in terms of time and space complexity, MDDE and the compared evolutionary multitasking algorithms do not show significant differences. In Fig. 9, the space consump-

tion of MDDE and the compared evolutionary multitasking algorithms is shown. Each point in the figure indicates the space cost by the algorithm in the corresponding test case. As shown in the figure, the positions of points in the same test case are quite similar, indicating that these algorithms' space consumption is similar. During the optimization of these algorithms, most of the space is occupied by the records in the tackled databases, which can also verify the above analysis regarding the space complexity.

7.5 Impact of proposed components

The proposed algorithm's new components contain the MDF, SBA operator, and PBM operator. To verify these components' effectiveness, three DE variants are implemented and compared with the original DE algorithm. These variants are listed as follows:

- (1) DE-MDF: In this variant, the MDF is embedded into the DE algorithm.
- (2) DE-SBA: The SBA operator is adopted in the DE algorithm.
- (3) DE-PBM: The PBM operator is embedded into the DE algorithm.

Table 6 shows the comparisons of experimental results where the best results are highlighted in **boldface**. To be specific, DE-MDF can achieve the best results in 5 test cases; DE-SBA can outperform DE in 10 test cases; and DE-PBM achieves the best result in the left 1 test case. When com-

Table 6 Impact of the proposed components

Test cases	DE		DE-MDF		DE-SBA		DE-PBM	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
T_1	1.69E+02	4.92E+01	2.20E+02	8.08E+01	2.67E+02	1.11E+02	2.20E+02	5.39E+01
T_2	6.11E+02	1.96E+02	8.10E+02	2.23E+02	7.77E+02	2.52E+02	7.04E+02	2.29E+02
T_3	1.35E+03	3.71E+02	1.51E+03	3.80E+02	1.66E+03	3.02E+02	1.54E+03	2.83E+02
T_4	2.22E+03	3.36E+02	2.67E+03	4.82E+02	3.19E+03	5.87E+02	2.46E+03	3.51E+02
T_5	3.18E+01	2.30E+01	5.14E+01	4.05E+01	3.07E+01	1.84E+01	3.00E+01	1.46E+01
T_6	3.16E+02	1.33E+02	5.07E+02	2.06E+02	5.00E+02	1.67E+02	3.55E+02	1.23E+02
T_7	6.49E+02	2.21E+02	9.25E+02	3.36E+02	1.05E+03	3.77E+02	7.59E+02	1.73E+02
T_8	1.24E+03	4.50E+02	1.26E+03	5.05E+02	1.60E+03	4.91E+02	1.49E+03	5.12E+02
T_9	3.07E+02	1.11E+02	3.00E+02	1.12E+02	3.39E+02	1.13E+02	2.95E+02	8.43E+01
T_{10}	8.97E+02	2.44E+02	1.19E+03	2.59E+02	1.10E+03	2.13E+02	9.69E+02	2.49E+02
T_{11}	1.82E+03	4.41E+02	2.18E+03	4.98E+02	3.11E+03	5.53E+02	2.17E+03	2.26E+02
T_{12}	2.96E+03	4.95E+02	3.59E+03	7.89E+02	3.93E+03	6.31E+02	3.49E+03	3.93E+02
T_{13}	1.18E+02	5.24E+01	1.94E+02	6.62E+01	1.79E+02	6.62E+01	1.37E+02	5.78E+01
T_{14}	5.44E+02	1.45E+02	7.29E+02	2.71E+02	7.64E+02	2.45E+02	5.93E+02	1.89E+02
T_{15}	1.11E+03	3.72E+02	1.07E+03	3.21E+02	1.23E+03	3.50E+02	1.25E+03	3.49E+02
T_{16}	1.81E+03	4.64E+02	1.88E+03	4.95E+02	2.24E+03	4.57E+02	1.92E+03	4.20E+02
+ / = / -	-		14/0/2		15/0/1		14/0/2	

paring DE and DE-MDF, the impact of the proposed MDF is verified. In all the compared 16 test cases, DE-MDF can outperform DE in 14 test cases, which indicates that the proposed MDF can help improve the population diversity and exploration search ability. Compared with DE, DE-SBA can achieve better performance in 15 test cases, which verifies that the SBA operator can improve the effectiveness of information exchange during the evolution. Correspondingly, the comparison between DE and DE-PBM shows the advantage of the PBM operator. With the help of the PBM operator, DE can outperform DE in 14 test cases. Overall, all the proposed components (i.e., MDF, SBA, and PBM) can contribute to the overall performance of MDDE.

7.6 Speedup ratio

In the MDDE algorithm, the MDF component is designed for two features. The first feature is to improve the efficiency in information exchange, whose impact has been verified in the last section. The second feature is to reduce the running time, which is based on the implementation of the distributed computation. To investigate the impact of the MDF component on the running time, the speedup ratio of the MDDE is measured. The speedup ratio is calculated by dividing the distributed algorithm's running time into the sequential algorithm's running time. A distributed algorithm with a higher speedup ratio indicates it can achieve higher parallel efficiency and scalability.

In the MDDE algorithm, each subpopulation is allocated to a single computation core, and each subpopulation evolves independently. The number of subpopulations in MDDE indicates its parallel granularity. The MDDE algorithm without the MDF is regarded as the sequential algorithm, and the MDDE algorithm with MDF is regarded as the distributed version. The running time of these two algorithms is measured, and the corresponding speedup ratios are calculated.

In Table 7, the running time of two algorithms and the speedup ratios in 16 test cases are listed. In all the test cases, the proposed MDDE algorithm can achieve speedup ratios higher than 3.3, which indicates the distributed framework of the MDDE algorithm can provide effective computational acceleration. The speedup ratios in different test cases vary. This is because different test cases are of different complexity and need different evaluation time. In test cases such as T_{16} , the speedup ratios of MDDE are higher. This is because these test cases are of higher complexity.

7.7 Parameter analysis

In MDDE, three parameters, i.e., subpopulation size SPS, crossover rate Cr, and migration interval MI , are utilized. These three parameters can directly affect the performance of MDDE and thus need to be carefully set.

When the value of SPS is low, the search of MDDE becomes relatively exploitative. In this case, the evolutionary information in the individuals is quickly exchanged among individuals, and the population diversity decreases. In con-

Table 7 Speedup ratios achieved by MDDE algorithm with MDF

Test cases	Without MDF	With MDF	Speedup ratio
	Running time (ms)	Running time (ms)	
T_1	2480.03	732.34	3.39
T_2	10,215.37	2922.67	3.50
T_3	23,696.21	6613.13	3.58
T_4	42,993.42	12,072.87	3.56
T_5	2697.92	791.31	3.41
T_6	11,138.55	3196.47	3.48
T_7	25,898.08	7274.01	3.56
T_8	46,841.2	13,063.72	3.59
T_9	2449.3	693.96	3.53
T_{10}	10,079.42	2850.96	3.54
T_{11}	23,406.72	6412.77	3.65
T_{12}	42,937.07	11,703.63	3.67
T_{13}	2624.25	767.18	3.42
T_{14}	11,019.55	3110.82	3.54
T_{15}	25,087.8	7083.4	3.54
T_{16}	45,262.05	12,605.41	3.59

Table 8 Parameters of MDDE and the corresponding recommended ranges

Parameters	Low	High	Recommended range
SPS	Exploitative	Exploratory	5–15
Cr	Exploitative	Exploratory	0.3–0.7
MI	Exploitative	Exploratory	5–25

trast, the search of MDDE tends to be exploratory, which is due to the high population diversity achieved by the large subpopulations. If the value of Cr is low, the evolutionary information in the mutant individual is less likely to be introduced in the trial individual. Thus, the search of MDDE becomes exploitative. On the contrary, the search of MDDE tends to be exploratory, which is due to the high population diversity achieved by the high crossover rate. If the value of MI is low, the evolutionary information identified by the subpopulations is exchanged frequently, and the search of MDDE tends to be exploitative. On the contrary, the population in MDDE with high MI maintains high diversity. Thus, the search of MDDE is exploratory. In Table 8, the impact of these three parameters on the MDDE algorithm is listed. Moreover, the recommended ranges of these parameters are listed, which are identified based on the prior experiments.

To identify which value combination of SPS and Cr can achieve the best performance in the given test cases, the average ranks of MDDE variants with different values of SPS and Cr are calculated. As shown in Fig. 10, the average ranks achieved by all the variants are between 4.5 and 6. There is no significant difference between these average ranks when the SPS and Cr values are defined in the given range. Moreover,

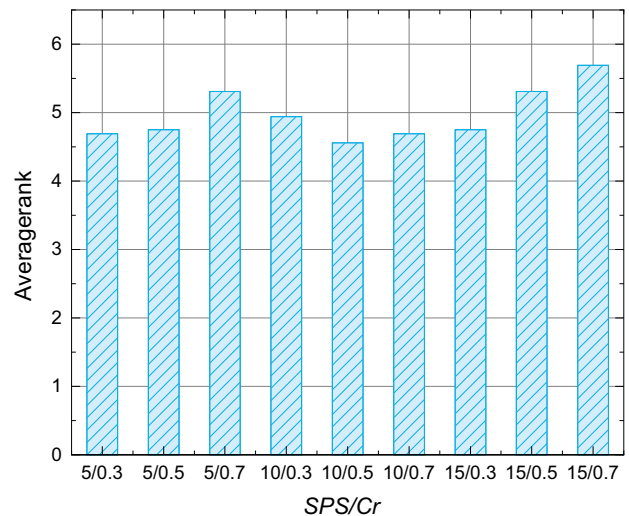


Fig. 10 Average ranks of MDDE variants with different value combinations of SPS and Cr

in Fig. 11, the average ranks of five MDDE variants adopting different values of MI are plotted. As shown in this figure, the average ranks achieved by these variants are between 2.5 and 3.5. To sum up, the parameter combination adopted in this paper (i.e., SPS = 10, Cr = 0.5, MI = 10) can achieve the best performance.

8 Conclusion

The multitasking database fragmentation problem is defined in this paper. For this problem, the MDDE algorithm is introduced, which includes MDF and two operators (i.e., SBA and

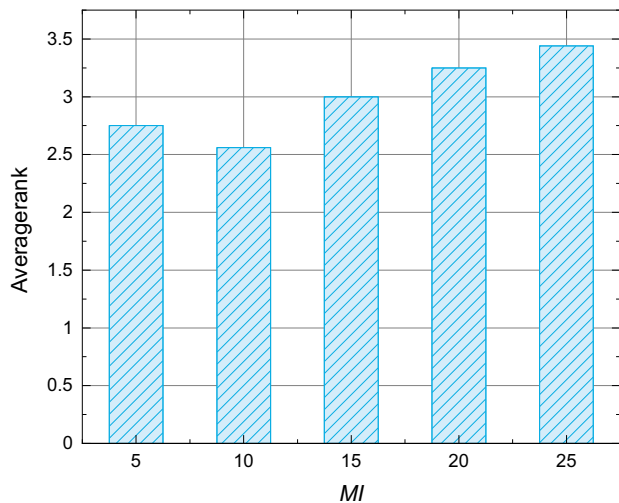


Fig. 11 Average ranks of MDDE variants with different values of MI

PBM). MDF is effective in exchanging information among different database fragmentation problems. The SBA operator can help adjust the fragment orders in different solutions. The PBM operator with adaptive mutation strategy selection is designed to sufficiently exchange allocation information in the individuals. Experimental results verify the advantages of the proposed algorithm in optimization performance. In the future, we can further investigate the approximation guarantee of the proposed algorithm. Furthermore, considering the effectiveness of the proposed MDF, we will apply it in other multitasking optimization problems.

References

- Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: a distributed architecture for secure database services. In: 2005 CIDR Conference (2005)
- Attasena, V., Darmont, J., Harbi, N.: Secret sharing for cloud data security: a survey. *VLDB J.* **26**(5), 657–681 (2017). <https://doi.org/10.1007/s00778-017-0470-9>
- Ciriani, V., Di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: *European Symposium on Research in Computer Security*, pp. 171–186. Springer (2007)
- Ciriani, V., Vimercati, S.D.C.D., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. *ACM Trans. Inf. Syst. Secur.* **13**(3), 1–33 (2010)
- De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Loose associations to increase utility in data publishing. *J. Comput. Secur.* **23**(1), 59–88 (2015)
- Feng, L., Huang, Y., Zhou, L., Zhong, J., Gupta, A., Tang, K., Tan, K.C.: Explicit evolutionary multitasking for combinatorial optimization: a case study on capacitated vehicle routing problem. *IEEE Trans. Cybern.* (2020). <https://doi.org/10.1109/tcyb.2019.2962865>
- Gao, J., Yu, J.X., Jin, R., Zhou, J., Wang, T., Yang, D.: Outsourcing shortest distance computing with privacy protection. *VLDB J.* **22**(4), 543–559 (2013). <https://doi.org/10.1007/s00778-012-0304-8>
- Gao, Z., Pan, Z., Zuo, C., Gao, J., Xu, Z.: An optimized deep network representation of multimutation differential evolution and its application in seismic inversion. *IEEE Trans. Geosci. Remote Sens.* **57**(7), 4720–4734 (2019). <https://doi.org/10.1109/tgrs.2019.2892567>
- Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York (1979)
- Ge, Y.F., Yu, W.J., Lin, Y., Gong, Y.J., Zhan, Z.H., Chen, W.N., Zhang, J.: Distributed differential evolution based on adaptive merge and split for large-scale optimization. *IEEE Trans. Cybern.* **48**(7), 2166–2180 (2018). <https://doi.org/10.1109/tycb.2017.2728725>
- Ge, Y.F., Yu, W.J., Zhan, Z.H., Zhang, J.: Competition-based distributed differential evolution. In: 2018 IEEE Congress on Evolutionary Computation. IEEE (2018). <https://doi.org/10.1109/cec.2018.8477758>
- Ge, Y.F., Cao, J., Wang, H., Zhang, Y., Chen, Z.: Distributed differential evolution for anonymity-driven vertical fragmentation in outsourced data storage. In: 2020 International Conference on Web Information Systems Engineering, pp. 213–226 (2020)
- Ge, Y.F., Yu, W.J., Cao, J., Wang, H., Zhan, Z.H., Zhang, Y., Zhang, J.: Distributed memetic algorithm for outsourced database fragmentation. *IEEE Trans. Cybern.* (2020). <https://doi.org/10.1109/tycb.2020.3027962>
- Gupta, A., Ong, Y.S., Feng, L.: Multifactorial evolution: toward evolutionary multitasking. *IEEE Trans. Evol. Comput.* **20**(3), 343–357 (2016). <https://doi.org/10.1109/tevc.2015.2458037>
- Hore, B., Mehrotra, S., Canim, M., Kantarcioglu, M.: Secure multidimensional range queries over outsourced data. *VLDB J.* **21**(3), 333–358 (2011). <https://doi.org/10.1007/s00778-011-0245-7>
- Köhler, J., Jünemann, K., Hartenstein, H.: Confidential database-as-a-service approaches: taxonomy and survey. *J. Cloud Comput.* **4**(1), 1–14 (2015)
- Li, J., Yao, W., Zhang, Y., Qian, H., Han, J.: Flexible and fine-grained attribute-based data storage in cloud computing. *IEEE Trans. Serv. Comput.* **10**(5), 785–796 (2017). <https://doi.org/10.1109/tsc.2016.2520932>
- Liaw, R.T., Ting, C.K.: Evolutionary many-tasking based on biocoenosis through symbiosis: a framework and benchmark problems. In: 2017 IEEE Congress on Evolutionary Computation, pp. 2266–2273. IEEE (2017). <https://doi.org/10.1109/cec.2017.7969579>
- Liaw, R.T., Ting, C.K.: Evolutionary manytasking optimization based on symbiosis in biocoenosis. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4295–4303 (2019). <https://doi.org/10.1609/aaai.v33i01.33014295>
- Price, K., Storn, R.M., Lampinen, J.A.: *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin (2006)
- Price, K.V.: Differential evolution. In: *Handbook of Optimization*, pp. 187–214 (2013)
- Rani, K., Sagar, R.K.: Enhanced data storage security in cloud environment using encryption, compression and splitting technique. In: 2017 International Conference on Telecommunication and Networks. IEEE (2017). <https://doi.org/10.1109/tel-net.2017.8343557>
- UbaidurRahman, N.H., Balamurugan, C., Mariappan, R.: A novel DNA computing based encryption and decryption algorithm. *Proc. Comput. Sci.* **46**, 463–475 (2015)
- Wang, Y., Yan, Z., Feng, W., Liu, S.: Privacy protection in mobile crowd sensing: a survey. *World Wide Web* **23**(1), 421–452 (2019). <https://doi.org/10.1007/s11280-019-00745-2>

25. Xu, X., Xiong, L., Liu, J.: Database fragmentation with confidentiality constraints: a graph search approach. In: 2015 ACM Conference on Data and Application Security and Privacy, pp. 263–270 (2015)
26. Yu, J., Wang, G., Mu, Y., Gao, W.: An efficient generic framework for three-factor authentication with provably secure instantiation. *IEEE Trans. Inf. Forensics Secur.* **9**(12), 2302–2313 (2014)
27. Yu, Y., Au, M.H., Ateniese, G., Huang, X., Susilo, W., Dai, Y., Min, G.: Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. *IEEE Trans. Inf. Forensics Secur.* **12**(4), 767–778 (2017)
28. Zheng, L.M., Zhang, S.X., Zheng, S.Y., Pan, Y.M.: Differential evolution algorithm with two-step subpopulation strategy and its application in microwave circuit designs. *IEEE Trans. Ind. Inform.* **12**(3), 911–923 (2016). <https://doi.org/10.1109/tii.2016.2535347>
29. Zhou, X.G., Peng, C.X., Liu, J., Zhang, Y., Zhang, G.J.: Underestimation-assisted global-local cooperative differential evolution and the application to protein structure prediction. *IEEE Trans. Evol. Comput.* **24**(3), 536–550 (2019). <https://doi.org/10.1109/tevc.2019.2938531>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.