Data Science and Engineering (2021) 6:380–391
https://doi.org/10.1007/s41019-021-00170-4

Set-Based Adaptive Distributed Differential Evolution for Anonymity-Driven Database Fragmentation

Yong-Feng Ge[1] · Jinli Cao[1] · Hua Wang[2] · Zhenxiang Chen[3] · Yanchun Zhang[2]

Received: 19 April 2021 / Revised: 21 July 2021 / Accepted: 12 August 2021 / Published online: 21 August 2021

Abstract
By breaking sensitive associations between attributes, database fragmentation can protect the privacy of outsourced data storage. Database fragmentation algorithms need prior knowledge of sensitive associations in the tackled database and set it as the optimization objective. Thus, the effectiveness of these algorithms is limited by prior knowledge. Inspired by the anonymity degree measurement in anonymity techniques such as *k*-anonymity, an anonymity-driven database fragmentation problem is defined in this paper. For this problem, a set-based adaptive distributed differential evolution (S-ADDE) algorithm is proposed. S-ADDE adopts an island model to maintain population diversity. Two set-based operators, i.e., set-based mutation and set-based crossover, are designed in which the continuous domain in the traditional differential evolution is transferred to the discrete domain in the anonymity-driven database fragmentation problem. Moreover, in the set-based mutation operator, each individual's mutation strategy is adaptively selected according to the performance. The experimental results demonstrate that the proposed S-ADDE is significantly better than the compared approaches. The effectiveness of the proposed operators is verified.

Keywords Differential evolution · Database fragmentation · Data privacy

1 Introduction

Outsourced data storage has shown its advantages in scalability and costs [13, 14, 33, 35]. Data security and privacy are still concerns when the organizations adopt the outsourced data storage [20, 26, 27, 30, 31]. In health industry, such concerns are significant [5, 11, 21, 25]. Although database encryption [12, 28] can protect the database privacy, encryption and decryption reduce the query efficiency accordingly [17, 24, 32], which is also crucial in outsourced data storage. By dividing attributes of sensitive associations, separate data models can protect the database privacy while not requiring time-consuming data transformation.

Several algorithms have been proposed for privacy protection in database fragmentation [15, 34]. In [2], encrypted database fragmentation was proposed to break the sensitive associations between attributes. Authors in [3] proved that the database fragmentation problem is NP-hard and proposed two heuristic strategies to achieve the optimal number of fragments. Then, a graph search algorithm [29] was proposed based on the confidentiality constraints. These algorithms are designed to address the database fragmentation problem with predefined privacy constraints. However, when it comes to the database fragmentation problem without prior knowledge, these algorithms become inapplicable. On the other side, anonymity techniques [16, 23] are also effective in database privacy protection. In these techniques, the performance is evaluated by the anonymity degree. A database with a higher anonymity degree is more likely to protect its data privacy. Unlike the privacy constraints set in database fragmentation algorithms, the anonymity degree can be calculated in all kinds of databases and does not require any prior knowledge. If we set the anonymity degree as the optimization objective, the database fragmentation algorithm can be used in all kinds of databases, including those lacking prior knowledge. Based on the above

✉ Yong-Feng Ge
yfge93@gmail.com

[1] Department of Computer Science and Information Technology, La Trobe University, Melbourne 3083, Australia

[2] Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne 3011, Australia

[3] School of Information Science and Engineering, University of Jinan, Jinan 250022, China

consideration, an anonymity-driven database fragmentation problem is defined.

For NP-hard optimization problems, various differential evolution (DE) algorithms have been proposed [7, 8, 18, 19]. They have shown the advantages in efficiency and reliability in optimization problems, including seismic inversion [6], microwave circuit design [36], and protein structure prediction [38]. In DE, different mutation strategies can address search problems of different properties. A proper design of mutation and crossover operators helps achieve the trade-off between exploration and exploitation. Considering the effectiveness of previous DE algorithms in the applications, it is worthy of applying the DE algorithm to the anonymity-driven database fragmentation problem.

In this paper, a set-based adaptive distributed differential evolution (S-ADDE) algorithm is proposed to address the anonymity-driven database fragmentation problem. The individual in S-ADDE represents the database fragmentation solution, and the anonymity degree of each solution is set as the fitness value of the individual. The update of the individuals in S-ADDE reflects the increase in anonymity degree in database fragmentation. Moreover, the contributions of this paper are listed as follows:

1. In order to maintain population diversity, we utilize an island model including four sub-populations;
2. Two set-based operators are proposed to transfer the continuous domain in traditional DE to discrete domain in the database fragmentation problem;
3. In the set-based mutation operator, each individual's mutation strategy is adaptively selected according to the evolving performance.

The remainder of this paper is organized as follows. Section 2 outlines the related work of database fragmentation for privacy protection, anonymity techniques, and recent DE applications. Afterward, the problem definition of the anonymity-driven database fragmentation problem is given in detail. In Sect. 4, a brief description of DE operators is given. Subsequently, we illustrate the proposed S-ADDE algorithm. In Sect. 6, extensive experiments and simulations are carried out, and the results are analyzed. Finally, in Sect. 7, we summarize this paper.

2 Related Work

Database fragmentation for data privacy protection was firstly proposed in [1]. In this work, all the database attributes are divided into two groups, which limits the performance in complex problems. Encrypted database fragmentation was introduced in [2], which has shown its advantage in breaking the sensitive association between attributes. Authors in [3] proved that the database fragmentation problem is NP-hard. Furthermore, two heuristic strategies were proposed to achieve the optimal number of fragments. Based on the confidentiality constraints in database fragmentation, a graph-based algorithm was proposed in [29], in which the nodes in the graph represent the database fragmentation solutions. In [4], the authors investigated the effectiveness of loose association in database fragmentation. The evolutionary algorithm was also utilized in database fragmentation. In [10], 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. These algorithms are designed based on predefined privacy constraints. When facing a database fragmentation problem laking of prior knowledge, these approaches are inapplicable.

On the other side, various anonymity techniques involving k-anonymity [23], l-diversity [16] were proposed for protecting the database privacy. The performance of the anonymity techniques is measured by the anonymity degree. One advantage of anonymity degree measurement is it does not require prior knowledge of the tackled database. Based on this advantage, the anonymity degree measurement can also be utilized in the database fragmentation problem.

In the last decades, DE algorithms have been widely utilized in actual applications and have shown their efficiency and reliability advantages. In [38], an underestimation-assisted global-local cooperative DE was proposed to enhance the effectiveness and the efficiency in optimization. The proposed algorithm was utilized in protein structure prediction and outperformed the competitors. Authors in [37] proposed a self-adaptive DE algorithm for addressing the batch-processing machine scheduling problem, in which the mutation operators and control parameter values are adaptively adjusted. In [22], a bi-objective elite DE was designed to optimize the multivalued logic networks. Two objective functions were used to simultaneously evaluate the fitness of the individuals in DE. In these applications, DE was redesigned according to the difficulty in the corresponding optimization problems.

3 Problem Definition

For a given relation schema R, a fragmentation $\mathcal{F}$ is legal if:

1. $\forall r \in R : \exists F \in \mathcal{F}$ such that $r \in F$ (a fragmentation cover all attributes)
2. $\forall F_i, F_j \in \mathcal{F}, i \neq j : F_i \cap F_j = \emptyset$ (fragments in the fragmentation do not have attributes in common)

For anonymity-driven database fragmentation, the objective is to identify a solution that can achieve the highest

anonymity degree for relation schema R. Anonymity degree of fragmentation is calculated as:

$$anonymity(\mathcal{F}) = min(anonymity(\forall F_i \in \mathcal{F})) \tag{1}$$

where $anonymity(\mathcal{F})$ is the anonymity degree of fragmentation $\mathcal{F}$. Anonymity degree of fragmentation is decided by its fragment of lowest anonymity degree.

Suppose fragmentation $\mathcal{F}$ is the optimal solution for R, it should meet the following conditions:

1. $\forall r \in R : \exists F \in \mathcal{F}$ such that $r \in F$
2. $\forall F_i, F_j \in \mathcal{F}, i \neq j : F_i \cap F_j = \varnothing$
3. $\forall \mathcal{F}\simeq anonymity(\mathcal{F}\simeq) \leq anonymity(\mathcal{F})$

which means fragmentation $\mathcal{F}$ is of higher anonymity degree than all the other legal solution that can satisfy the first two conditions.

4 Differential Evolution (DE)

DE process includes three operators, i.e., mutation, crossover, and selection. An elementary description of these three operators is given as follows.

4.1 Mutation

The mutation operator utilizes the difference between individuals in the population to construct the mutant individuals. $\mathbf{v}_i^g$ represents the ith mutant individual at generation g. Various mutation strategies have been proposed and listed as follows:

DE/rand/1

$$\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \cdot (\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g) \tag{2}$$

DE/current-to-best/1

$$\mathbf{v}_i^g = \mathbf{x}_i^g + F \cdot (\mathbf{x}_{best}^g - \mathbf{x}_i^g) + F \cdot (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) \tag{3}$$

DE/best/1

$$\mathbf{v}_i^g = \mathbf{x}_{best}^g + F \cdot (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) \tag{4}$$

DE/best/2

$$\mathbf{v}_i^g = \mathbf{x}_{best}^g + F \cdot (\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) + F \cdot (\mathbf{x}_{r3}^g - \mathbf{x}_{r4}^g) \tag{5}$$

DE/rand/2

$$\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \cdot (\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g) + F \cdot (\mathbf{x}_{r4}^g - \mathbf{x}_{r5}^g) \tag{6}$$

where $\mathbf{x}_{best}^g$ indicates the best individual at generation g; $r1$, $r2$, $r3$, $r4$, and $r5$ are indexes of five random individuals; F is the differential factor.

4.2 Crossover

In this process, the mutant individual $\mathbf{v}_i^g$ exchanges evolutionary information with the current individual $\mathbf{x}_i^g$ and generates a trial individual $\mathbf{u}_i^g$, which is formulated as:

$$\mathbf{u}_{i,j}^g = \begin{cases} \mathbf{v}_{i,j}^g, & if\ rand(0,1) \leq Cr\ or\ j = j_{rand} \\ \mathbf{x}_{i,j}^g, & otherwise \end{cases} \tag{7}$$

where $rand(0, 1)$ is a random float number between 0 and 1; j_{rand} represents a random integer to guarantee at least one bit of $\mathbf{u}_i^g$ comes from $\mathbf{v}_i^g$; Cr represents the crossover rate.

4.3 Selection

In this stage, the fitness values of the current individual and the trial individual are compared. The individual with higher fitness value is selected and kept. For a minimization problem $f(x)$, the selection process is formulated as follows:

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g, & if\ f(\mathbf{u}_i^g) \leq f(\mathbf{x}_i^g) \\ \mathbf{x}_i^g, & otherwise \end{cases} \tag{8}$$

where $\mathbf{x}_i^{g+1}$ is the target individual at the next generation.

When utilizing DE to solve the database fragmentation problem, DE's individuals can represent solutions of database fragmentation. The fitness values of the individuals represent the anonymity degrees of database fragmentation solutions. Thus, the improvement of fitness values in DE can help achieve better database fragmentation.

5 Set-Based Adaptive Distributed Differential Evolution (S-ADDE)

In this section, the proposed S-ADDE algorithm is described in detail. The representation manner of the S-ADDE algorithm is first introduced. Then, the island model of S-ADDE is introduced. Afterward, a set-based mutation operator, an adaptive strategy of mutation strategy selection, and a set-based crossover operator are proposed. Finally, to illustrate the overall S-ADDE process, the pseudo-code of S-ADDE is given and explained.

5.1 Representation

In Fig. 1, a sample database is given, which contains nine attributes and six records. As shown in the figure, the database is divided into three fragments. These three fragments make up a fragmentation solution shown at the bottom of the figure.

ID	Area (A)	YoB (B)	Gender (C)	Race (D)	ZIP (E)	LoS (F)	Type (G)	Disease (H)
1	Western NY	1965	Male	Black	2141	4	Urgent	Fever
2	Central NY	1965	Male	White	2155	2	Emergency	Chest pain
3	Finger Lakes	1965	Female	Black	2167	1	Elective	Back pain
4	Western NY	1964	Female	White	2142	1	Emergency	Vomiting
5	Finger Lakes	1964	Female	Black	2161	5	Elective	Fever
6	Central NY	1964	Female	White	2152	7	Urgent	Chest pain

ID	Area (A)	YoB (B)	ZIP (E)
1	Western NY	1965	2141
2	Central NY	1965	2155
3	Finger Lakes	1965	2167
4	Western NY	1964	2142
5	Finger Lakes	1964	2161
6	Central NY	1964	2152

ID	Gender (C)	Race (D)	LoS (F)
1	Male	Black	4
2	Male	White	2
3	Female	Black	1
4	Female	White	1
5	Female	Black	5
6	Female	White	7

ID	Type (G)	Disease (H)
1	Urgent	Fever
2	Emergency	Chest pain
3	Elective	Back pain
4	Emergency	Vomiting
5	Elective	Fever
6	Urgent	Chest pain

F_1	Area (A)	YoB (B)	ZIP (E)
F_2	Gender (C)	Race (D)	LoS (F)
F_3	Type (G)	Disease (H)	

Fig. 1 An example of the individual representation in the S-ADDE algorithm, in which the sample database is divided into three fragments

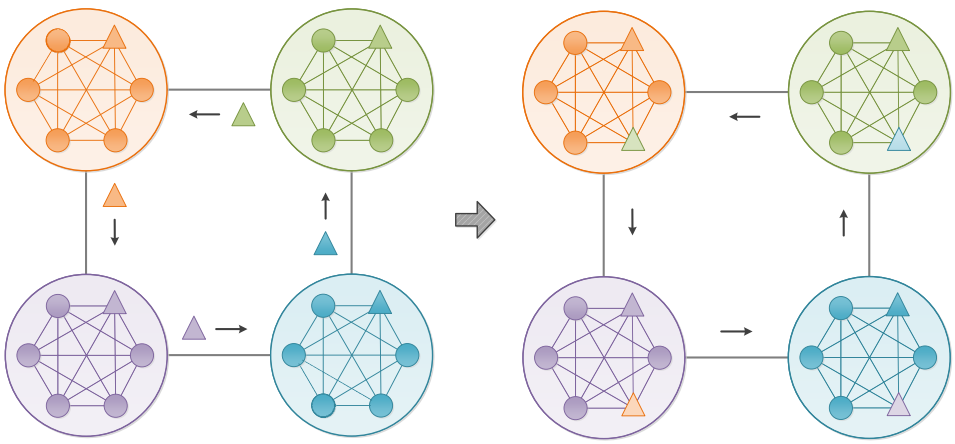

Fig. 2 An example of island model in S-ADDE

Each individual in the proposed S-ADDE algorithm represents a database fragmentation solution. Accordingly, each bit in the individual indicates one attribute in the database, and its value indicates which fragment the corresponding attribute is chosen for allocation.

5.2 Island Model

DE algorithm with the distributed framework has shown its advantages in optimization performance and speed. One classic distributed framework of the DE algorithm is the island model. The DE algorithm population is divided into several sub-populations in the island model, and each sub-population evolves independently. According to the communication topology, sub-populations share their elite individuals with a given interval, which is referred to as the migration interval. Once one sub-population receives the migrated elite individuals, individuals in the current generation are randomly selected and replaced.

In the proposed S-ADDE algorithm, an island model with a ring communication topology is utilized. An example of the island model is given in Fig. 2. As shown in the example, each big circle indicates a sub-population. In the big circles, small triangles and circles represent the best individuals and

the other sub-population individuals. The best individuals in sub-populations are sent to the neighborhood sub-populations on the communication topology with the predefined migration interval. Afterward, one individual in each sub-population is chosen by random and replaced by the received elite individual.

By dividing the entire population of the DE algorithm into several sub-populations with independent evolution, the island model can help the S-ADDE algorithm maintain the population diversity. By migrating elite individuals in the sub-populations, the island model can enhance the S-ADDE algorithm's population quality. If the migration operator is appropriately executed, the S-ADDE algorithm can achieve the trade-off between exploration and exploitation. Moreover, since each sub-population evolves independently, the island model can be directly implemented in a distributed manner, which is crucial for speedup in evolution.

5.3 Set-Based Mutation

As mentioned above, the traditional DE algorithm's mutation strategies are designed to optimize the continuous optimization problems. In our tackled database fragmentation problem, the fragmentation solutions are of the discrete domain. One challenge of mutation strategy design is how to transfer the continuous domain in traditional mutation strategies to the discrete domain in the database fragmentation problem. In the S-ADDE algorithm, a set-based mutation operator is proposed. To be specific, the fragment solution in the individual of S-ADDE is regarded as a set. Thus, the calculation between two individuals is transferred to the set calculation between two database fragmentation solutions. In DE mutation operators, two kinds of calculation rules are involved. The first part is to calculate the difference between two individuals. The second part is to calculate the sum of two individuals. When calculating the difference and the sum for each bit, if the values of two individuals are the same, the same value is kept. If the values of two individuals are different, both two values are kept as a set. With the help of the above calculation rules, all the existing mutation strategies can be utilized in discrete optimization problems.

As mentioned above, when tackling discrete optimization problems, traditional mutation strategies cannot be directly utilized. In the proposed set-based mutation operator, the calculation between two individuals is transferred to the set calculation between the elements in the individuals. When utilizing these two calculation rules in the traditional mutation strategies, all the mutation strategies can be utilized. For instance, in DE/rand/1 mutation strategy, the difference between two randomly selected individuals is firstly calculated and then added to the third individual. Accordingly, the elements only contained by the first random individual are extracted and added to the fragments in the third random individual.

5.4 Adaptive Mutation Strategy Selection

In the set-based mutation operator, the calculation between individuals in the mutation strategy is transferred to the set-based calculation between fragmentation solutions. The set-based calculation can fit all the mutation strategies and achieve its effect. Every mutation strategy has a unique effect on optimization. For example, DE/rand/1 strategy can enhance the population strategy and outperform the multi-modal search problems. In contrast, the DE/best/1 strategy can enhance the DE algorithm's exploitation ability and achieve high performance in single-modal search problems. An adaptive strategy for selecting a proper mutation strategy during the evolution is designed based on this background. All the chosen mutation strategies are inserted into a strategy pool, and the mutation strategy of the each individual is adaptively selected.

At the beginning of the evolution, each individual chooses one mutation strategy from the strategy pool. After executing the chosen mutation strategy, if the generated individual is kept in the population, which indicates higher competitiveness, the current mutation strategy will be utilized in the next generation. Otherwise, a mutation strategy is randomly chosen from the strategy pool, and the current mutation strategy is replaced.

If a mutation strategy can fit a search problem well, it is more likely to generate better individuals and of higher probability to be utilized. On the contrary, a mutation strategy that cannot achieve ideal performance is less likely to be utilized. In various search problems, the probability of executing every mutation strategy is adaptively adjusted, which makes the proposed S-ADDE outperform.

5.5 Set-Based Crossover

After the set-based mutation operator, the chosen individuals' evolutionary information is extracted and used to construct a mutant individual. In the mutant individual, some bits may contain more than one element. According to the above problem definition, in a legal solution for database fragmentation, each element must be involved in one fragment and only appear once, which indicates a fragmentation covering all attributes, and the fragments do not have attributes in common. To make the mutant individual from the set-based mutation operator legal, if one bit contains two or more elements, one element is chosen by random. Afterward, the legal mutant individual $\mathbf{v}_i$ exchanges the evolutionary information with the current individual $\mathbf{x}_i$. For each element, with the crossover rate, it is assigned in the same fragment as individual $\mathbf{v}_i$.

With the proposed set-based crossover operator's help, fragmentation information from the mutant individual and the current individual is exchanged. Since the new individual's redundant elements are removed, for each element, it is only contained by one fragment and will not appear more than once in the generated individual. Randomly, an individual containing fragmentation information from two individuals is generated. If the generated individual is better than the current individual, it will be kept, and the competitiveness of the entire population is improved.

5.6 Overall Process

The pseudo-code of S-ADDE is given in Algorithm 1. As shown in the pseudo-code, a master-slave model is utilized to implement the S-ADDE algorithm. At the master node, the entire population is divided into *N* sub-populations and sent to the corresponding *N* slave nodes. With the predefined migration interval *MI*, the master node receives the migrated individuals from the slave nodes and sends these elite individuals to the corresponding slave nodes. The

Algorithm 1 Pseudo-code of S-ADDE

```
1: procedure MASTER NODE
2:     Set initial generation g = 0
3:     Divide the population into N sub-populations
4:     while terminal condition is not met do
5:         if g % MI = 0 then
6:             Receive elite individuals from slave nodes
7:             Send the elite individuals to corresponding slave nodes
8:         end if
9:         g = g + 1
10:    end while
11:    Output the best solution
12: end procedure
13:
14: procedure SLAVE NODE
15:    for each individual do
16:        Randomly choose a mutation strategy from the pool
17:    end for
18:    for every generation do
19:        for each individual do
20:            Perform the set-based mutation operator
21:            Perform the set-based crossover operator
22:            Perform the selection operator
23:            if the generated individual is of better then
24:                Replace the current individual by the generated individual
25:            else
26:                Randomly choose a mutation strategy from the pool
27:            end if
28:        end for
29:        if g % MI = 0 then
30:            Send the best individual to the master node
31:            Receive the elite individual from the master node
32:            Replace a randomly chosen individual
33:        end if
34:    end for
35: end procedure
```

Table 1 Properties of 16 test cases

Test cases	*NR*	*NSR*	*NA*	*NS*
T_1	100,000	1000	15	3
T_2	100,000	2000	15	3
T_3	100,000	3000	15	3
T_4	100,000	4000	15	3
T_5	200,000	1000	20	4
T_6	200,000	2000	20	4
T_7	200,000	3000	20	4
T_8	200,000	4000	20	4
T_9	300,000	1000	25	5
T_{10}	300,000	2000	25	5
T_{11}	300,000	3000	25	5
T_{12}	300,000	4000	25	5
T_{13}	400,000	1000	30	6
T_{14}	400,000	2000	30	6
T_{15}	400,000	3000	30	6
T_{16}	400,000	4000	30	6

migration process is executed until the terminal condition is satisfied. Finally, the fragmentation solutions of the sub-populations are collected, and the best fragmentation solution is outputted.

At the slave node, each sub-population evolves independently. At the beginning of the evolution, a mutation strategy is randomly selected from the mutation strategy pool for each individual. For each individual in the sub-population, the set-based mutation operator is firstly carried out, and the corresponding mutant individual is generated. Then, the mutant individual is transferred to be legal in the set-based crossover operator and exchanges the current individual's evolutionary information. If the newly generated individual is more competitive, it is kept in the population, and the current mutation strategy is kept. Otherwise, the current individual in the population is kept, and a mutation strategy is randomly selected from the mutation strategy pool. With the given migration interval *MI*, the slave node sends its best individual to the master node and receives one elite individual from the master node. The migrated elite individual replaces one randomly chosen individual in the current population.

6 Experimental Result

6.1 Experimental Setup

In the experiments, 16 test cases are utilized to verify the effectiveness of the proposed S-ADDE algorithm in database fragmentation. These test cases are generated based on the public datasets of New York State Department of Health.[1] Table 1 outlines the properties of these test cases, which include numbers of records *NR*, numbers of sample records *NSR*, numbers of attributes *NA*, and numbers of sites *NS*.

In the proposed S-ADDE algorithm, the number of sub-populations in S-ADDE is set as 4; the sub-population size

Table 2 Comparisons with competitive approaches

Test cases	HA		DE		S-DDE		S-ADDE	
	Avg	SD	Avg	SD	Avg	SD	Avg	SD
T_1	1.25E+00	1.68E−01	2.02E+00	3.01E−01	**2.38E+00**	**9.26E−02**	2.34E+00	1.18E−01
T_2	1.46E+00	2.83E−01	2.72E+00	3.78E−01	**3.14E+00**	**9.89E−02**	3.08E+00	2.17E−01
T_3	1.67E+00	3.30E−01	3.08E+00	4.71E−01	**3.87E+00**	**1.75E−01**	3.84E+00	1.54E−01
T_4	1.89E+00	4.75E−01	3.67E+00	5.52E−01	**4.46E+00**	**2.70E−01**	4.43E+00	2.07E−01
T_5	1.31E+00	2.32E−01	1.88E+00	1.87E−01	2.13E+00	2.60E−01	**2.15E+00**	**2.45E−01**
T_6	1.49E+00	2.74E−01	2.31E+00	3.07E−01	2.59E+00	4.32E−01	**2.81E+00**	**4.13E−01**
T_7	1.52E+00	3.43E−01	2.71E+00	5.21E−01	3.09E+00	4.42E−01	**3.29E+00**	**4.59E−01**
T_8	1.55E+00	3.81E−01	2.96E+00	4.94E−01	3.46E+00	4.84E−01	**3.69E+00**	**7.01E−01**
T_9	1.41E+00	3.22E−01	1.60E+00	1.34E−01	2.08E+00	3.12E−01	**2.19E+00**	**2.92E−01**
T_{10}	1.55E+00	3.48E−01	2.02E+00	3.35E−01	2.62E+00	4.50E−01	**2.99E+00**	**5.24E−01**
T_{11}	1.85E+00	3.98E−01	2.26E+00	2.80E−01	3.05E+00	4.79E−01	**3.17E+00**	**5.00E−01**
T_{12}	1.96E+00	6.20E−01	2.64E+00	4.62E−01	3.59E+00	5.45E−01	**4.31E+00**	**9.20E−01**
T_{13}	1.09E+00	6.37E−02	1.19E+00	2.97E−02	1.37E+00	1.28E−01	**1.45E+00**	**1.26E−01**
T_{14}	1.14E+00	8.38E−02	1.32E+00	6.93E−02	1.68E+00	2.33E−01	**1.72E+00**	**2.48E−01**
T_{15}	1.23E+00	1.28E−01	1.40E+00	1.37E−01	1.82E+00	3.43E−01	**1.83E+00**	**2.71E−01**
T_{16}	1.34E+00	1.59E−01	1.47E+00	1.41E−01	1.96E+00	3.56E−01	**2.08E+00**	**2.78E−01**

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

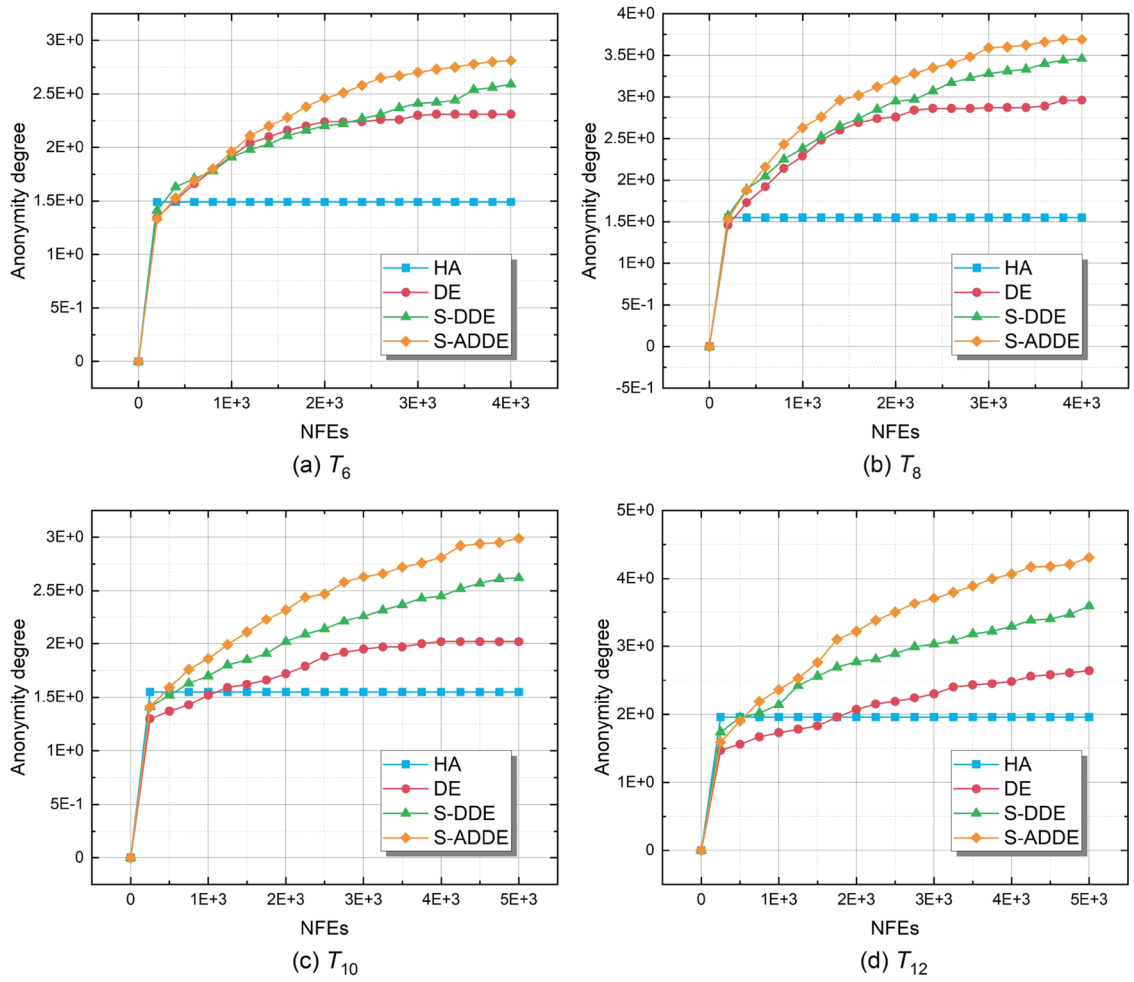

Fig. 3 Convergence curves of competitive approaches on four typical test cases

SPS is set as 10; the migration interval *MI* is set as 10; the crossover rate *Cr* is set as 0.5. For all the algorithms, the maximum fitness evaluation number is set as $NS \times 10^3$.

The island model of S-ADDE is implemented by the Message Passing Interface (MPI). Each sub-population is assigned to a computation core in the CPU. The communication between sub-populations is implemented by the message passing between CPU cores.

6.2 Comparisons with Competitive Approaches

To verify the effectiveness of the proposed S-ADDE algorithm on test cases, two competitive approaches, i.e., heuristic algorithm [3] and differential evolution [19], are compared. The specific description of these two competitive approaches are listed as follows:

1. HA [3]: This is a state-of-the-art heuristic algorithm for database fragmentation problem, in which two heuristic strategies are designed to achieve the optimal fragmentation.
2. DE [19]: This approach acts as a baseline algorithm. The differences between DE and S-ADDE in performance show the effectiveness of the proposed operators.
3. S-DDE [9]: In this algorithm, the database fragmentation problem is optimized by set-based mutation and crossover operators. Different from the proposed S-ADDE, its mutation strategy is not adaptively selected.

Furthermore, on each test case, the proposed S-ADDE and the compared approaches are performed in 25 independent runs.

The average and standard deviation values obtained by all the approaches are calculated and listed in Table 2. The best results are labeled in **boldface**. S-ADDE algorithm can outperform the other approaches on all the test cases. Due to the exploration ability, S-ADDE can achieve a better performance than HA. In HA, the search direction is led by the predefined heuristic strategy and its search direction

Table 3 Effectiveness of adaptive mutation strategy selection

Test cases	S-ADDE-rand		S-ADDE-current-to-best		S-ADDE-best		S-ADDE	
	Avg	SD	Avg	SD	Avg	SD	Avg	SD
T_1	2.29E+00	1.81E−01	2.20E+00	2.18E−01	2.20E+00	2.27E−01	**2.34E+00**	**1.18E−01**
T_2	3.07E+00	1.72E−01	**3.10E+00**	**1.71E−01**	2.95E+00	2.91E−01	3.08E+00	2.17E−01
T_3	3.72E+00	3.14E−01	**3.84E+00**	**2.06E−01**	3.55E+00	3.76E−01	**3.84E+00**	**1.54E−01**
T_4	4.34E+00	2.71E−01	4.30E+00	4.45E−01	4.16E+00	4.64E−01	**4.43E+00**	**2.07E−01**
T_5	2.09E+00	2.50E−01	2.07E+00	2.32E−01	1.94E+00	2.61E−01	**2.15E+00**	**2.45E−01**
T_6	2.55E+00	3.19E−01	2.67E+00	4.23E−01	2.42E+00	3.13E−01	**2.81E+00**	**4.13E−01**
T_7	3.23E+00	5.08E−01	**3.29E+00**	**4.53E−01**	2.89E+00	4.78E−01	**3.29E+00**	**4.59E−01**
T_8	3.57E+00	5.97E−01	3.57E+00	5.66E−01	3.13E+00	5.73E−01	**3.69E+00**	**7.01E−01**
T_9	2.15E+00	3.03E−01	2.11E+00	1.88E−01	1.90E+00	2.97E−01	**2.19E+00**	**2.92E−01**
T_{10}	2.83E+00	5.26E−01	2.93E+00	5.49E−01	2.50E+00	4.04E−01	**2.99E+00**	**5.24E−01**
T_{11}	3.22E+00	4.90E−01	**3.46E+00**	**6.15E−01**	3.00E+00	5.92E−01	3.17E+00	5.00E−01
T_{12}	3.89E+00	7.25E−01	3.96E+00	8.01E−01	3.56E+00	6.24E−01	**4.31E+00**	**9.20E−01**
T_{13}	1.42E+00	1.02E−01	**1.48E+00**	**1.20E−01**	1.37E+00	1.08E−01	1.45E+00	1.26E−01
T_{14}	1.69E+00	2.34E−01	1.68E+00	2.70E−01	1.56E+00	1.47E−01	**1.72E+00**	**2.48E−01**
T_{15}	1.81E+00	2.29E−01	1.81E+00	2.31E−01	1.72E+00	2.55E−01	**1.83E+00**	**2.71E−01**
T_{16}	1.93E+00	2.10E−01	**2.08E+00**	**3.72E−01**	1.72E+00	2.33E−01	**2.08E+00**	**2.78E−01**

is narrow. In the complex test cases such as T_{14} and T_{16}, S-ADDE is more likely to get trapped by the local optimum. Compared with DE's results, we can verify the island model and proposed set-based operators are effective in S-ADDE. Firstly, the island model maintains the population diversity, which is helpful in complex test cases. Secondly, the individuals' evolutionary information is effectively extracted by the set-based operators and used to reconstruct offsprings. When comparing with S-DDE, the advantage of the adaptive mutation strategy selection in S-ADDE is verified. With the help of the proposed adaptive strategy, S-ADDE can achieve a better balance between the exploratory and exploitative search. Therefore, S-ADDE can outperform S-DDE on 12 out of 16 test cases.

The obtained convergence curves on four typical test cases are plotted in Fig. 3. Three lines indicate the convergence curves obtained by HA, DE, and S-ADDE, 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 optimum and stagnates. Thanks to the exploration ability of DE and S-ADDE, they can continuously improve the anonymity degree during the search. The differences between the green lines of S-ADDE and the red lines of DE verify the effectiveness of the island model and the proposed set-based operators in S-ADDE. Moreover, the proposed adaptive strategy can help accelerate the convergence speed of the proposed S-ADDE. Compared with S-DDE, we can find that S-ADDE can achieve higher values of anonymity degree during the entire optimization process. Overall, S-ADDE can achieve the highest convergence speed due to the trade-off between exploration and exploitation.

Based on the definition of the database fragmentation problem, the time complexity of calculating the anonymity degree for a given fragmentation is $\mathcal{O}(NA \times NR \times \log NR)$. Thus, the time complexity of the proposed S-ADDE algorithm is $\mathcal{O}(NA \times NR \times \log NR \times NS)$. According to the original papers, the time complexity of HA is $\mathcal{O}(NA^2 \times NR \times \log NR)$. The time complexity of DE and S-DDE is $\mathcal{O}(NA \times NR \times \log NR \times NS)$, which is the same as S-ADDE. Overall, in terms of the time complexity, S-ADDE and all three compared algorithms do not show significant differences. Moreover, since S-DDE and S-ADDE are implemented based on the distributed framework, the parallel computation can help reduce the running time.

6.3 Effect of Adaptive Mutation Strategy Selection

In S-ADDE, the mutation strategy is adaptively selected. An experiment is carried out to verify its effect. Besides the proposed S-ADDE algorithms, three variants of S-ADDE adopting different mutation strategies are implemented and listed as follows:

1. *S-ADDE-rand* This variant adopts DE/rand/1 as the mutation strategy. The mutation strategy is not changed during the entire evolution.

Table 4 Effect of S-ADDE Results on Original Datasets

Test cases	Without S-ADDE	With S-ADDE		
	AD	min(AD)	avg(AD)	max(AD)
T_1	1.11	23.6	33.14	44.53
T_2	1.11	24.69	34.69	47.76
T_3	1.11	26.07	37.44	55.58
T_4	1.11	25.3	37.53	56.29
T_5	1.1	28.53	60.01	105.76
T_6	1.1	29.86	57.22	99.64
T_7	1.1	29.05	88.47	236.31
T_8	1.1	29.67	68.07	148.2
T_9	1.09	36.19	132.26	352.93
T_{10}	1.09	44.35	96.91	179.91
T_{11}	1.09	33.4	146.69	463.48
T_{12}	1.09	47.59	151.22	435.56
T_{13}	1	11.64	69.51	265.12
T_{14}	1	12.68	72.29	251.5
T_{15}	1	12.83	78.55	306.98
T_{16}	1	14.17	80.23	299.83

2. *S-ADDE-current-to-best* This variant utilizes DE/current-to-best/1 mutation strategy.
3. *S-ADDE-best* This variant adopts DE/best/1 as the mutation strategy.

To be noted, except for the adaptive mutation strategy selection part, all the compared variants share the same distributed island model and operators as the original S-ADDE.

In Table 3, the average and standard deviation obtained by variants and the original S-ADDE are listed. The best results are labeled in **boldface**. In total, S-ADDE can outperform on 13 test cases; S-ADDE-current-to-best can outperform on 6 test cases; S-ADDE-rand and S-ADDE-best cannot outperform on any test case. Compared with S-ADDE-current-to-best, although it can outperform 6 test cases, the proposed S-ADDE can achieve the same results on 3 test cases. Compared with the other two variants, i.e., S-ADDE-rand and S-ADDE-best, the proposed S-ADDE shows its advantages in all the test cases. Although these two variants do not outperform in the comparison, they are effective during the cooperation with other mutation strategies. This point can be verified by the performance of S-ADDE, which utilizes the adaptive mutation strategy selection.

6.4 Effect of S-ADDE Results on Original Datasets

In this section, the anonymity degrees of original datasets without database fragmentation and with database fragmentation are compared. As shown in Table 4, *AD* indicates the anonymity degree of each dataset. Values *min*(*AD*), *avg*(*AD*), and *max*(*AD*) represent the minimal, average, and maximum values of anonymity degree obtained by the fragments in S-ADDE.

According to Table 4, it is clear that S-ADDE can afford significant enhancement to original datasets in anonymity. Without S-ADDE, the anonymity degree of original datasets is around 1, which means each record in the original datasets can be directly identified according to its characteristics. With the help of the S-ADDE database fragmentation result, the anonymity degree of the original dataset in each fragment increases. Focusing on the column of *min*(*AD*), which indicates the minimal anonymity degree obtained by all the fragments, it is obvious that the privacy

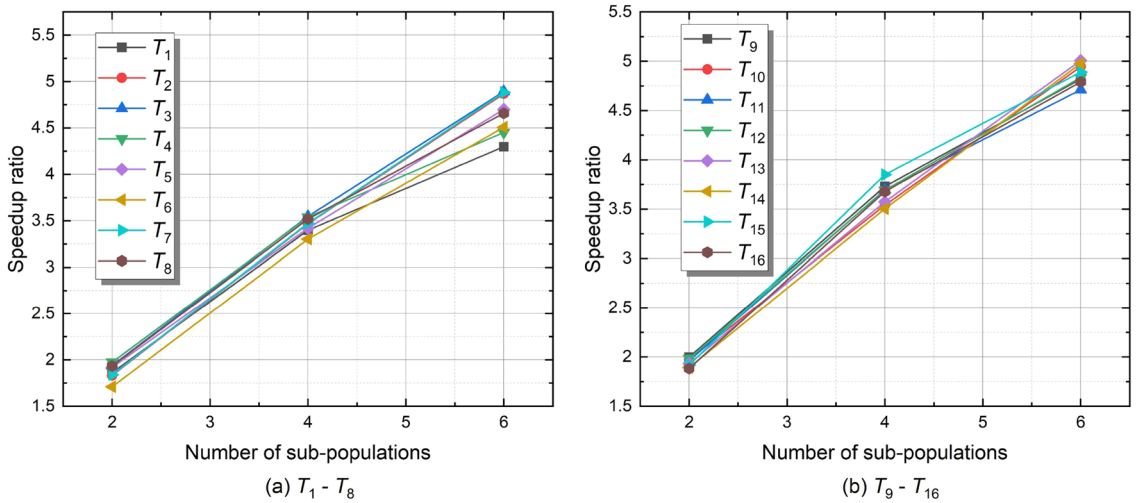

Fig. 4 Speedup ratios of S-ADDE on all test cases

of records is protected. In T_1, the value of $min(AD)$ is 23.6. This value means each record shares the same characteristics with the other 22.6 records and cannot be directly identified. Thus, the privacy of datasets is guaranteed.

6.5 Speedup Ratio

In the distributed algorithm, the speedup ratio is an important indicator. It can reflect the computation efficiency of the distributed algorithm. The speedup ratio is calculated by dividing the running time taken by the distributed algorithm into the sequential algorithm's running time. A distributed algorithm with a higher speedup ratio can achieve higher distributed computation efficiency, which is crucial in maintaining the algorithm's scalability.

In the proposed S-ADDE algorithm, each sub-population is allocated to a single computation core, and each sub-population evolves independently. Thus, the number of sub-populations in S-ADDE directly reflects its parallel granularity. S-ADDE algorithms' running time with different numbers of sub-populations (1, 2, 4, 6) is measured. The S-ADDE algorithm with a single sub-population is regarded as the sequential algorithm, and the S-ADDE algorithms with multiple sub-populations are regarded as the distributed algorithms.

In Fig. 4, the speedup ratios of S-ADDE on 16 test cases are plotted. The speedup ratios significantly increase with the parallel granularity of S-ADDE increases from two to six. The speedup ratio curves in different test cases vary. This is because different test cases are of different complexity and need different evaluation time. In general, the communication time of S-ADDE on different test cases does not have a significant difference. Thus, a test case of higher evaluation time, such as T_{15} and T_{16}, can help achieve speedup ratios. When adopted in actual optimization problems, which contains higher complexity, the proposed S-ADDE algorithm can further show its scalability and speed advantages.

7 Conclusion

An anonymity-driven database fragmentation problem has been defined in this paper. To address this problem, we have proposed the S-ADDE algorithm. S-ADDE algorithm utilizes an island model to improve the population diversity, which is crucial in complex search problems. Moreover, we have proposed two set-based operators, i.e., set-based mutation operator with adaptive mutation strategy selection and set-based crossover operator. According to the analysis of the experimental results, the proposed S-ADDE algorithm is significantly better than the compared algorithms. The computation efficiency of S-ADDE has been investigated and the effectiveness of the proposed operators has been verified.

In this paper, the privacy issue (i.e., anonymity degree) of the database fragmentation is optimized. Furthermore, the utility issue (e.g., communication cost) of the database fragmentation can be further investigated and optimized in future work. In addition, considering the effectiveness of the proposed set-based operators and the adaptive selection strategy, we can further apply them in the other discrete optimization problems, such as the database allocation problem.

References

1. Aggarwal G, Bawa M, Ganesan P, Garcia-Molina H, Kenthapadi K, Motwani R, Srivastava U, Thomas D, Xu Y (2005) Two can keep a secret: a distributed architecture for secure database services. In: 2005 CIDR conference
2. Ciriani V, Di Vimercati SDC, Foresti S, Jajodia S, Paraboschi S, Samarati P (2007) Fragmentation and encryption to enforce privacy in data storage. In: European symposium on research in computer security, Springer, pp 171–186
3. Ciriani V, Vimercati SDCD, Foresti S, Jajodia S, Paraboschi S, Samarati P (2010) Combining fragmentation and encryption to protect privacy in data storage. ACM Trans Inf Syst Secur 13(3):1–33
4. De Capitani, di Vimercati S, Foresti S, Jajodia S, Livraga G, Paraboschi S, Samarati P (2015) Loose associations to increase utility in data publishing. J Comput Secur 23(1):59–88
5. Du J, Michalska S, Subramani S, Wang H, Zhang Y (2019) Neural attention with character embeddings for hay fever detection from twitter. Health Inf Sci Syst 7(1):1–7. https://doi.org/10.1007/s13755-019-0084-2
6. Gao Z, Pan Z, Zuo C, Gao J, Xu Z (2019) An optimized deep network representation of multimutation differential evolution and its application in seismic inversion. IEEE Trans Geosci Remote Sens 57(7):4720–4734. https://doi.org/10.1109/tgrs.2019.2892567
7. Ge YF, Yu WJ, Zhang J (2016) Diversity-based multi-population differential evolution for large-scale optimization. In: 2016 genetic and evolutionary computation conference, ACM Press, pp 31–32. https://doi.org/10.1145/2908961.2908995
8. Ge YF, Yu WJ, Lin Y, Gong YJ, Zhan ZH, Chen WN, Zhang J (2018) Distributed differential evolution based on adaptive mergence and split for large-scale optimization. IEEE Trans Cybern 48(7):2166–2180. https://doi.org/10.1109/tcyb.2017.2728725

9. Ge YF, Cao J, Wang H, Zhang Y, Chen Z (2020a) Distributed differential evolution for anonymity-driven vertical fragmentation in outsourced data storage. In: 2020 Web information systems engineering, Springer International Publishing, pp 213–226
10. Ge YF, Yu WJ, Cao J, Wang H, Zhan ZH, Zhang Y, Zhang J (2020) Distributed memetic algorithm for outsourced database fragmentation. IEEE Trans Cybern. https://doi.org/10.1109/tcyb.2020.3027962
11. Islam MR, Kabir MA, Ahmed A, Kamal ARM, Wang H, Ulhaq A (2018) Depression detection from social network data using machine learning techniques. Health Inf Sci Syst 6(1):1–12. https://doi.org/10.1007/s13755-018-0046-0
12. Köhler J, Jünemann K, Hartenstein H (2015) Confidential database-as-a-service approaches: taxonomy and survey. J Cloud Comput 4(1):1–14
13. Li J, Yao W, Zhang Y, Qian H, Han J (2017) Flexible and fine-grained attribute-based data storage in cloud computing. IEEE Trans Serv Comput 10(5):785–796. https://doi.org/10.1109/tsc.2016.2520932
14. Li M, Sun X, Wang H, Zhang Y, Zhang J (2011) Privacy-aware access control with trust management in web service. World Wide Web 14(4):407–430. https://doi.org/10.1007/s11280-011-0114-8
15. Lin X, Orlowska M, Zhang Y (1993) A graph based cluster approach for vertical partitioning in database design. Data Knowl Eng 11(2):151–169. https://doi.org/10.1016/0169-023x(93)90003-8
16. Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) L-diversity: privacy beyond k-anonymity. In: 22nd international conference on data engineering. IEEE. https://doi.org/10.1109/icde.2006.1
17. Peng M, Zeng G, Sun Z, Huang J, Wang H, Tian G (2017) Personalized app recommendation based on app permissions. World Wide Web 21(1):89–104. https://doi.org/10.1007/s11280-017-0456-y
18. Price K, Storn RM, Lampinen JA (2006) Differential evolution: a practical approach to global optimization. Springer Science & Business Media
19. Price KV (2013) Differential evolution. Handbook of optimization. Springer, Berlin, pp 187–214
20. Rani K, Sagar RK (2017) Enhanced data storage security in cloud environment using encryption, compression and splitting technique. In: 2017 2nd international conference on telecommunication and networks (TEL-NET). IEEE. https://doi.org/10.1109/tel-net.2017.8343557
21. Sarki R, Ahmed K, Wang H, Zhang Y (2020) Automated detection of mild and multi-class diabetic eye diseases using deep learning. Health Inf Sci Syst 8(1):1–9. https://doi.org/10.1007/s13755-020-00125-5
22. Sun J, Gao S, Dai H, Cheng J, Zhou M, Wang J (2020) Bi-objective elite differential evolution algorithm for multivalued logic networks. IEEE Trans Cybern 50(1):233–246. https://doi.org/10.1109/tcyb.2018.2868493
23. Sweeney L (2002) K-anonymity: a model for protecting privacy. Int J Uncertainty Fuzziness Knowl Based Syst 10(05):557–570. https://doi.org/10.1142/s0218488502001648
24. UbaidurRahman NH, Balamurugan C, Mariappan R (2015) A novel DNA computing based encryption and decryption algorithm. Proc Comput Sci 46:463–475
25. Vimalachandran P, Liu H, Lin Y, Ji K, Wang H, Zhang Y (2020) Improving accessibility of the Australian my health records while preserving privacy and security of the system. Health Inf Sci Syst 8(1):1–9. https://doi.org/10.1007/s13755-020-00126-4
26. Wang H, Zhang Z, Taleb T (2018) Special issue on security and privacy of IOT. World Wide Web 21(1):1–6
27. Wang H, Wang Y, Taleb T, Jiang X (2020) Special issue on security and privacy in network computing. World Wide Web 23(2):951–957. https://doi.org/10.1007/s11280-019-00704-x
28. Wang Y, Yan Z, Feng W, Liu S (2019) Privacy protection in mobile crowd sensing: a survey. World Wide Web 23(1):421–452. https://doi.org/10.1007/s11280-019-00745-2
29. Xu X, Xiong L, Liu J (2015) Database fragmentation with confidentiality constraints: a graph search approach. In: 2015 ACM conference on data and application security and privacy, pp 263–270
30. Yu J, Wang G, Mu Y, Gao W (2014) An efficient generic framework for three-factor authentication with provably secure instantiation. IEEE Trans Inf Forensics Secur 9(12):2302–2313
31. Yu Y, Au MH, Ateniese G, Huang X, Susilo W, Dai Y, Min G (2017) Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans Inf Forensics Secur 12(4):767–778
32. Zhang F, Wang Y, Liu S, Wang H (2020) Decision-based evasion attacks on tree ensemble classifiers. World Wide Web. https://doi.org/10.1007/s11280-020-00813-y
33. Zhang J, Tao X, Wang H (2014) Outlier detection from large distributed databases. World Wide Web 17(4):539–568
34. Zhang Y, Orlowska ME (1994) On fragmentation approaches for distributed database design. Inf Sci Appl 1(3):117–132. https://doi.org/10.1016/1069-0115(94)90005-1
35. Zhang Y, Chen X, Li J, Wong DS, Li H, You I (2017) Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. Inf Sci 379:42–61. https://doi.org/10.1016/j.ins.2016.04.015
36. Zheng LM, Zhang SX, Zheng SY, Pan YM (2016) Differential evolution algorithm with two-step subpopulation strategy and its application in microwave circuit designs. IEEE Trans Ind Inf 12(3):911–923. https://doi.org/10.1109/tii.2016.2535347
37. Zhou S, Xing L, Zheng X, Du N, Wang L, Zhang Q (2019) A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times. IEEE Trans Cybern. https://doi.org/10.1109/tcyb.2019.2939219
38. Zhou XG, Peng CX, Liu J, Zhang Y, Zhang GJ (2019) Underestimation-assisted global-local cooperative differential evolution and the application to protein structure prediction. IEEE Trans Evol Comput 24(3):536–550. https://doi.org/10.1109/tevc.2019.2938531