A Benefit-Driven Genetic Algorithm for Balancing Privacy and Utility in Database Fragmentation

Yong-Feng Ge, Jinli Cao
Department of Computer Science and Information Technology
La Trobe University
Victoria, Australia

Hua Wang
Institute for Sustainable Industries and Liveable Cities
Victoria University
Victoria, Australia

Jiao Yin
Department of Computer Science and Information Technology
La Trobe University
Victoria, Australia

Wei-Jie Yu
(Corresponding Author)
School of Information Management
Sun Yat-sen University
Guangzhou, China
ywj21c@163.com

Zhi-Hui Zhan, Jun Zhang
School of Computer Science and Engineering
South China University of Technology
Guangzhou, China

ABSTRACT

In outsourcing data storage, privacy and utility are significant concerns. Techniques such as data encryption can protect the privacy of sensitive information but affect the efficiency of data usage accordingly. By splitting attributes of sensitive associations, database fragmentation can protect data privacy. In the meantime, data utility can be improved through grouping data of high affinity. In this paper, a benefit-driven genetic algorithm is proposed to achieve a better balance between privacy and utility for database fragmentation. To integrate useful fragmentation information in different solutions, a matching strategy is designed. Two benefit-driven operators for mutation and improvement are proposed to construct valuable fragments and rearrange elements. The experimental results show that the proposed benefit-driven genetic algorithm is competitive when compared with existing approaches in database fragmentation.

CCS CONCEPTS

• **Security and privacy** → **Information accountability and usage control**; • **Theory of computation** → **Random search heuristics**; • **Mathematics of computing** → *Combinatorial optimization*;

KEYWORDS

Database fragmentation, genetic algorithm, benefit-driven strategy

ACM Reference format:
Yong-Feng Ge, Jinli Cao, Hua Wang, Jiao Yin, Wei-Jie Yu, and Zhi-Hui Zhan, Jun Zhang. 2019. A Benefit-Driven Genetic Algorithm for Balancing Privacy and Utility in Database Fragmentation. In *Proceedings of Genetic and Evolutionary Computation Conference, Prague, Czech Republic, July 13–17, 2019 (GECCO '19),* 6 pages.
https://doi.org/10.1145/3321707.3321778

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321778

1 INTRODUCTION

Outsourcing data storage provided by commercial cloud service has shown its advantages in high convenience and low cost. However, confidentiality of sensitive information in outsourcing data storage is still a big concern [14, 19, 21, 22]. To be specific, data security and privacy protection issues are primary inhibitors in the adoption of cloud service [2]. The challenge of protecting the confidentiality of private data has attracted attention from both academic research and industrial community [17, 23]. For this purpose, many approaches have been proposed [8–10]. Traditional approaches for private data protection involve encryption [11]. Querying over encrypted data requires specific indexing techniques or decryption approaches [18], which directly affect the efficiency of data utilizing.

Fragmentation is on the other side of data encryption [16, 20]. By splitting attributes of data in different vertical fragments, sensitive associations among data can be protected [7, 12]. A significant advantage of data fragmentation over data encryption is its high convenience regarding data access and querying. The utility is also considerable concern in outsourcing data storage [1]. Having attributes of high affinity in the same fragments can improve the efficiency of data querying and evaluation. Fragmentation aims to balance the trade-off between privacy conservation and data utility. An ideal fragmentation over database can protect the confidentiality of data as well as satisfy utility requirements.

In the previous literature, several algorithms have been proposed for fragmentation optimization. In [4], the problem of optimizing fragmentation regarding constraints and affinity between attributes is defined. Also, two heuristic algorithms which can optimize the number of fragments and the sum of affinity values are proposed. Moreover, the authors prove that the identified problems are NP-hard. Publication of data in term of multiple loose associations between different pairs of fragments is studied in [5]. Given a specific level of protection for sensitive associations, the proposed

heuristic algorithm can provide satisfying fragmentation. Querying utility is also considered in different fragments. The genetic algorithm has shown its effectiveness in various fields due to high performance [6, 13]. In the problem of database fragmentation, the genetic algorithm has also been adopted. In [15], genetic algorithm is utilized to generate fragmentation which can satisfy the given constraints. However, there are only two fragments involved in this algorithm, and the utility of fragmentation is ignored. Also, the performance of the proposed genetic algorithm is not enhanced by specifically designed operators.

In this paper, to achieve a better balance between constraints and utility requirements in database fragmentation, a benefit-driven genetic algorithm is proposed. To integrate useful fragmentation information in different solutions, a matching-based recombination operator is proposed. Besides, to maintain population diversity as well as introduce useful fragmentation information, two mutation operators named random mutation and benefit-driven mutation are proposed. Furthermore, a benefit-driven improvement strategy is designed to merge fragments in generated individuals to improve the competitiveness of the solution.

The main contributions of this paper are listed as follows:

(1) An individual matching strategy is designed for recombination to maximize the effectiveness of recombination.
(2) Motivated by benefit degree of constructing fragments and rearranging elements, a benefit-driven mutation operator is proposed.
(3) A benefit-driven improvement strategy is utilized to improve to the competitiveness of generated solution.

The remainder of this paper is organized as follows. In Section 2, the problem of database fragmentation is defined. Section 3 describes the proposed benefit-driven algorithm in detail, which includes a novel matching strategy and benefit-driven operators. Extensive experiments with discussion are provided in Section 4. Finally, Section 5 concludes.

2 PROBLEM DEFINITION

The objective of database fragmentation problem is to find a fragmentation that can satisfy the given constraints as well as optimize utility. Let R be a relation schema, a set of well-defined constraints C over R and a weighted list of utility requirements $\mathcal{U}$, find a fragmentation $\mathcal{F}$ of R such that all the following conditions hold.

(1) $\forall F \in \mathcal{F}, \forall c \in C : c \nsubseteq F$;
(2) $\forall F_i, F_j \in \mathcal{F}, i \neq j : F_i \cap F_j = \emptyset$;
(3) $\forall F'$ satisfying the first two conditions such that $utility(\mathcal{F}') \leq utility(\mathcal{F})$

Suppose fragmentation $\mathcal{F}$ is the optimal solution. It means fragmentation $\mathcal{F}$ satisfied all the predefined constraints and no other fragmentation can provide higher utility as well as meet the given constraints.

3 BENEFIT-DRIVEN GENETIC ALGORITHM

3.1 Representation and Initialization

The first step in designing a genetic algorithm is to devise a suitable representation scheme. In this problem, a solution containing n elements can be coded as a vector whose length is n. In this representation, a bit in the vector is associated with the corresponding element. $Solution[i] = j$ indicates ith element is located in fragment j.

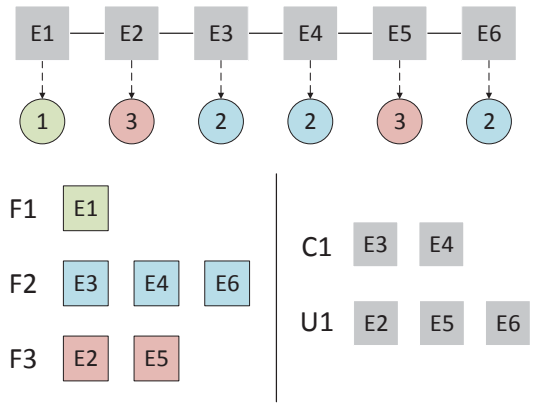

Figure 1: Illustration of representation.

Each individual in the initial population is generated according to the above representation. To increase the diversity of the initial population, each initial individual is created by random. This way, the initialized individuals are not only of high diversity but also feasible to the problem.

As shown in Figure 1, an individual containing six elements is initialized by random. Suppose there are three fragments, for each element, its initial fragment index is initialized between 1 and 3. Through initialization, the first fragment (F1) contains one element (E1); the second fragment (F2) includes three elements (E3, E4, and E6); the third fragment (F3) contains two elements (E2 and E5).

As an example, in Figure 1, constraint fitness of given solution is the sum of constraint fitness achieved by three fragments which is 1. With the same way, utility fitness of sample solution is the sum of utility fitness produced by three fragments which is 0.

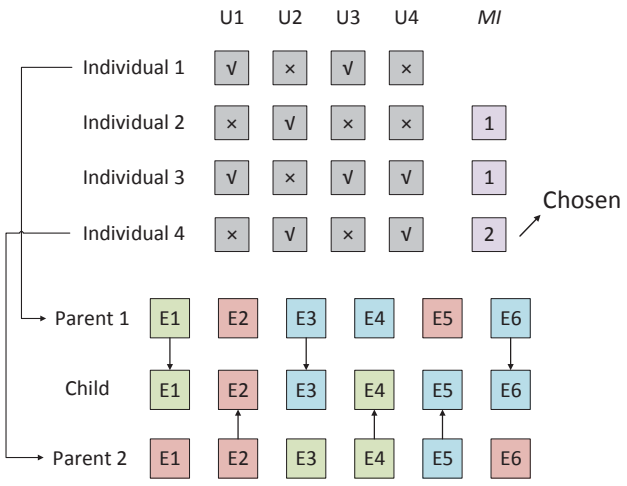

Figure 2: Illustration of recombination process.

3.2 Matching-Based Recombination (MBR)

Although it is improbable that the current population contains the optimal solution, it is possible that those solutions include several correctly arranged elements. According to the definition of constraint fitness and utility fitness, these solutions should show low constraint fitness and high utility fitness. One major goal of recombination is to extract the correctly arranged elements in parent individuals and integrate them into the generated child individual.

To achieve a better effect of recombination, a matching strategy is designed. Once the first parent individual in recombination operator is identified, it can be helpful if the other parent individual matches the chosen one. Based on this idea, the satisfaction condition of utility requirements achieved by individuals is considered. The goal of the matching strategy is to find the other parent individual who can help achieve more utility requirements.

Parameter matching index (MI) is designed to describe the matching degree between each pair of individuals. MI_i^j between individual i and individual j is calculated by counting the number of utility requirements which can be satisfied by individual j but not included in individual i. This way, once one of the parent individuals is identified in the recombination operator, the other parent individual who can supply helpful fragmentation information can be found. For individual i, its matching individual is the one who owns the highest value of MI.

As shown in Figure 2, individual 1 is the first chosen parent individual. Then, the matching index between individual 1 and other individuals is calculated. Individual 4 fulfills the second utility requirement (U2) and the fourth utility requirement (U4), which are not satisfied by individual 1. The matching index between individual 1 and individual 4 MI_1^4 is 2.

Once two parent individuals are determined, recombination of these two individuals is executed. To combine fragmentation information in parent individuals adequately, uniform crossover is adopted. As an example in Figure 2, in the child individual, fragment indexes of three elements are from one parent individual and the other three indexes are from the other parent individual.

3.3 Mutation

Mutation helps maintain population diversity as well as introduce additional information to prevent the algorithm from converging too fast and prematurely. In this problem, for each offspring generated by recombination operator, its fragmentation information is from parent individuals. An ideal offspring can integrate useful fragmentation information from parents. However, recombination operator is difficult to construct new fragmentation utility satisfaction. This way, in the generated population, the distribution of utility satisfaction is very likely to lose balance, which means some of the utility requirements are satisfied by most of the individuals while other utility requirements are not included in any individual.

To overcome the satisfaction biases appearing in population, in this algorithm, two mutation operators are proposed, namely, random mutation (RM) and benefit-driven mutation (BDM). With a completely random manner, RM can help introduce random fragmentation information to maintain population diversity. Furthermore, to adaptively introduce fragmentation information for specific utility requirement as well as decrease satisfaction biases, BDM

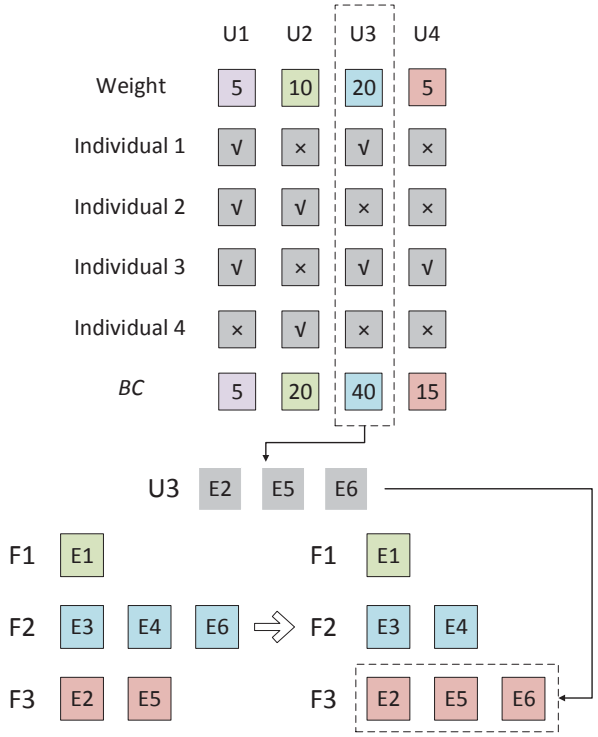

Figure 3: Illustration of adaptive mutation process.

is designed accordingly. When executing, two mutation operators are randomly selected. RM is selected with a predefined possibility P_m.

Random Mutation (RM). For each element in the given individual, it has a predefined possibility P_{rm} to be chosen. The chosen element is randomly assigned to a new fragment.

Benefit-Driven Mutation (BDM). To counter the utility satisfaction biases, the satisfaction degree of each utility is necessary to acquire. This degree can be indicated by the number of individuals who can satisfy it. For each utility requirement, its satisfaction degree should be of higher value if more utility requirements are satisfied. Furthermore, considering the weight of each utility requirement is different. The weight of utility requirement is also necessary to be utilized.

After calculating the benefit degree of each utility requirement, the utility requirement of highest benefit degree is chosen. For each individual mutated through BDM operator, if the selected utility requirement is not satisfied, elements in the corresponding utility requirement are adjusted to locate in the same fragment. This way, the utility requirement of the highest benefit is constructed, which may have never included in the population during the entire evolution process.

As shown in Figure 3, benefit index of each fragment is calculated based on satisfaction distribution of each utility requirement and its weight. From the given result, U3 is of the highest benefit index. Thus, elements E2, E5, and E6 are rearranged to fulfill U3.

Algorithm 1 Pseudo-code of benefit-driven genetic algorithm

```
Set G = 0 (G is current generation of population)
Initialize population P
Evaluate constraint and utility fitness values for each initial individual
while stopping criterion is not met do
    G = G + 1
    Update benefit value for each utility requirement
    Update benefit value for each element
    for each two individuals in population do
        Select the individual of higher fitness as the first parent individual
        Match the second parent individual
        Perform recombination operator on parent individuals
        if rand(0, 1) < P_m then
            Execute RM operator on offspring
        else
            Execute BDM operator on offpsring
        end if
        Perform BDIS on offspring
        Evaluate constraint and utility fitness values of offspring
        Use the generated offpsring to replace the individual of lower fitness
    end for
end while
Output convergence data
Output the best solution
```

3.4 Benefit-Driven Improvement Strategy (BDIS)

In the proposed recombination and mutation operators, individuals are enhanced through integrating and constructing accurately aligned fragments. Additionally, a benefit-driven improvement strategy is proposed, which is based on migrating elements that are likely to locate at wrong positions.

The proposed improvement strategy includes two basic procedures: one is to calculate the migration benefit of each element, and the other is element migration. With the help of the utility evaluation result, the benefit of migrating each component can be directly calculated.

Migration benefit of ith element BM_i is calculated as:

$$BM_i = \sum_{j=1}^{NU} DW_i^j \tag{1}$$

$$DW_i^j = \begin{cases} 0, & \text{if } Uj \text{ does not contain } Ei \\ -W_j, & \text{if } Uj \text{ containts } Ei \text{ and is satisfied} \\ W_j, & \text{otherwise} \end{cases} \tag{2}$$

where NU is the number of utility requirements, DW_j^i is directed weight of Ei on Uj and W_j is the weight of Uj.

The calculation result can help indicate the benefit degree of migrating elements. Thus, the element of highest migration benefit is chosen and randomly rearranged in another fragment.

3.5 Overall Process

The entire process of the proposed algorithm is shown in Algorithm 1. Firstly, the population is initialized, and each individual in the initial population is evaluated. Then, during each generation in the evolution process, benefit values for utility requirements and elements are updated. For every two individual in the population, the fitter individual is chosen as the first parent individual in recombination. Then the other individual is selected according to the proposed matching strategy. After recombination, the offspring is mutated and improved by BDIS operator. The generated offspring is used to replace the other individual that is of lower fitness in the two chosen individuals. Once the stopping criterion is met, the convergence data during the evolution process and the best solution are output.

Table 1: Properties of test cases

Test cases	NE	NC	SC	NU	SU
T_1	10	20	[2,6]	40	[4,8]
T_2	15	30	[2,8]	60	[4,10]
T_3	20	40	[2,11]	80	[4,13]
T_4	25	50	[2,13]	100	[4,15]
T_5	30	60	[2,16]	120	[4,18]
T_6	35	70	[2,18]	140	[4,20]
T_7	40	80	[2,21]	160	[4,23]
T_8	45	90	[2,23]	180	[4,25]
T_9	10	20	[4,8]	40	[4,8]
T_{10}	15	30	[4,10]	60	[4,10]
T_{11}	20	40	[4,13]	80	[4,13]
T_{12}	25	50	[4,15]	100	[4,15]
T_{13}	30	60	[4,18]	120	[4,18]
T_{14}	35	70	[4,20]	140	[4,20]
T_{15}	40	80	[4,23]	160	[4,23]
T_{16}	45	90	[4,25]	180	[4,25]

4 EXPERIMENTAL STUDIES

4.1 Experimental Setup

In the experiments, 16 test cases of different complexity are generated by random and carried out to evaluate the performance of the proposed GA-BD. Properties of test cases including the number of elements NE, number of constraints NC and corresponding size SC, number of utility requirements NU and corresponding size SU are listed in Table 1.

The parameter setting of the proposed algorithm is listed as follows. Population size NP is set as 30; the maximum number of fitness evaluations $MaxFEs$ is set as $NE \times 10^3$; mutation rate P_m is set as 0.9 and random mutation rate P_{rm} is set as 0.2.

Additionally, to improve the objectivity of experiments, all the GA based algorithms run 25 times independently. The proposed and all the compared algorithms are implemented by C++ and performed on an Intel Core i5-4278U CPU with 8GB RAM.

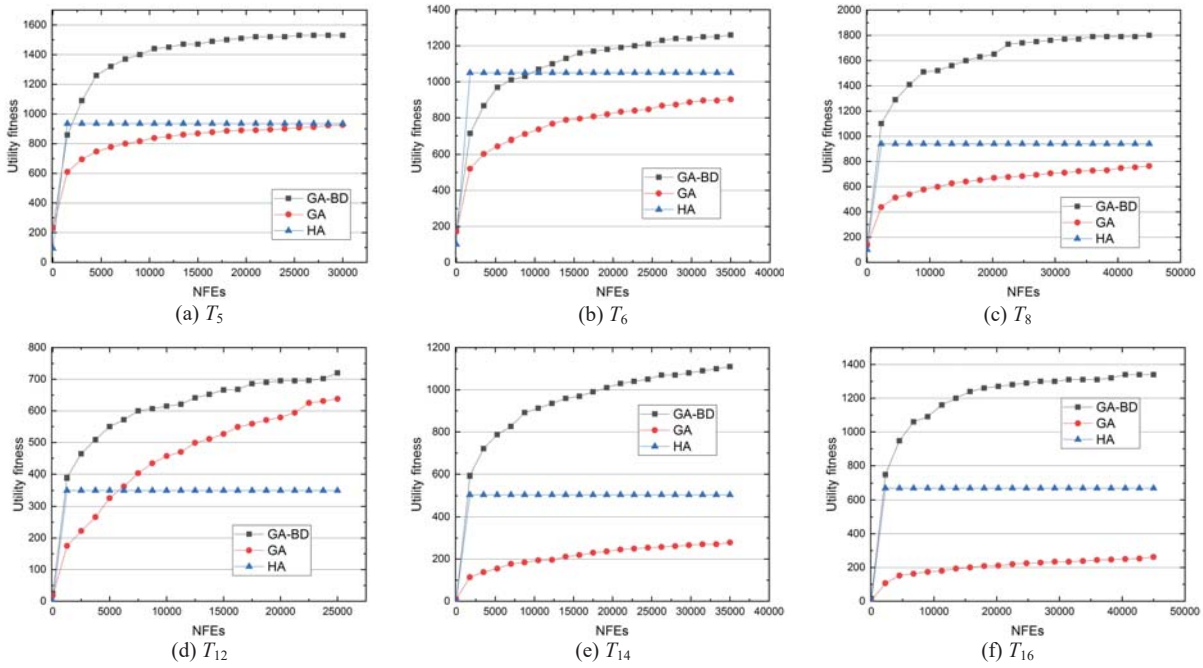

Figure 4: Comparisons of convergence performance between GA-BD and existing approaches on six typical test cases.

4.2 Comparisons with Existing Approaches

In this subsection, we compare the proposed GA-BD with two existing approaches to further reveal its advantage. These three algorithms are listed as follows:

(1) GA-BD: GA-BD is the complete version of the proposed algorithm, including MBR, BDM, and BDIS.
(2) GA [3]: Same as the previous subsection, GA is utilized for comparison.
(3) HA [4]: This heuristic algorithm is the state-of-the-art approach in handling constraint and utility in database fragmentation.

Table 2 shows the comparisons of experimental results where the best results are highlighted in boldface. From the result, it can be found that the proposed GA-BD algorithm can achieve the best results in 15 out of 16 test cases, which is higher than 4 and 0 of GA and HA, respectively. Our proposed GA-BD algorithm comprehensively outperforms the other algorithms regarding solution accuracy. Results of the significant test are also labeled in Table 2. GA-BD is significantly better than GA and HA on 12 and 16 test cases, while significantly worse than GA and HA on only 1 and 0 test cases, respectively.

Besides, Figure 4 shows the convergence curves of three approaches on six typical test cases. Each point on the plot is calculated by taking the average of 25 independent runs. These six test cases, namely, T_5, T_6, T_8, T_{12}, T_{14} and T_{16} are of different numbers of elements as well as different complexity.

Convergence curves of these six typical test cases indicate that GA-BD has the highest convergence rate among the three approaches. For these test cases, GA-BD shows both better exploration and exploitation abilities than the other two. To summarize, by utilizing benefit-driven strategies, the proposed GA-BD is a very promising algorithm for database fragmentation problem.

5 CONCLUSIONS

Overall, a benefit-driven genetic algorithm has been proposed in this paper to achieve the balance between privacy and utility in database fragmentation. An individual matching strategy is proposed and embedded in recombination operator to integrate useful fragmentation information in different individuals. Random mutation and benefit-driven mutation are designed to maintain population diversity as well as introduce valuable fragments. Moreover, to rearrange elements in fragments, a benefit-driven improvement strategy is proposed. Extensive experiments have been conducted, and the effectiveness of the proposed operators has been verified. Compared with existing approaches, our proposed GA-BD algorithm has shown its high competitiveness.

REFERENCES

[1] Piotr Berman and Sofya Raskhodnikova. 2014. Approximation Algorithms for Min-Max Generalization Problems. *ACM Transactions on Algorithms* 11, 11 (2014), 5.
[2] Deyan Chen and Hong Zhao. 2012. Data security and privacy protection issues in cloud computing. In *2012 International Conference on Computer Science and Electronics Engineering*, Vol. 1. IEEE, 647–651.
[3] PC Chu and John E Beasley. 1998. Constraint handling in genetic algorithms: the set partitioning problem. *Journal of Heuristics* 4, 4 (1998), 323–357.
[4] Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. 2010. Combining fragmentation and

Table 2: Comparisons between GA-BD and existing approaches

Approaches	GA-BD		GA			HA	
	Mean	Std	Mean	Std		Result	
T_1	**7.50E+02**	0.00E+00	**7.50E+02**	0.00E+00	≈	5.45E+02	−
T_2	7.71E+02	1.51E+01	**7.75E+02**	0.00E+00	+	3.70E+02	−
T_3	**7.54E+02**	5.73E+01	6.96E+02	3.68E+01	−	4.70E+02	−
T_4	**9.02E+02**	4.66E+01	7.69E+02	5.77E+01	−	5.85E+02	−
T_5	**1.53E+03**	2.16E+01	9.26E+02	4.02E+01	−	9.35E+02	−
T_6	**1.26E+03**	1.24E+02	9.02E+02	7.26E+01	−	1.05E+03	−
T_7	**1.29E+03**	1.33E+02	7.16E+02	3.60E+01	−	1.07E+03	−
T_8	**1.80E+03**	1.75E+02	7.64E+02	6.97E+01	−	9.40E+02	−
T_9	**5.15E+02**	0.00E+00	**5.15E+02**	0.00E+00	≈	3.05E+02	−
T_{10}	**7.75E+02**	0.00E+00	**7.75E+02**	0.00E+00	≈	1.50E+02	−
T_{11}	**8.82E+02**	3.82E+01	6.85E+02	9.83E+01	−	6.05E+02	−
T_{12}	**7.20E+02**	6.78E+01	6.38E+02	5.57E+01	−	3.50E+02	−
T_{13}	**9.35E+02**	6.22E+01	4.58E+02	8.83E+01	−	4.10E+02	−
T_{14}	**1.11E+03**	1.29E+02	2.78E+02	5.17E+01	−	5.05E+02	−
T_{15}	**1.06E+03**	1.26E+02	2.40E+02	4.17E+01	−	4.60E+02	−
T_{16}	**1.34E+03**	1.53E+02	2.62E+02	4.32E+01	−	6.70E+02	−
−/≈ /+			**12/3/1**			**16/0/0**	

encryption to protect privacy in data storage. *ACM Transactions on Information and System Security* 13, 3 (2010), 22.

[5] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Giovanni Livraga, Stefano Paraboschi, and Pierangela Samarati. 2015. Loose associations to increase utility in data publishing. *Journal of Computer Security* 23, 1 (2015), 59–88.

[6] Dunwei Gong, Jing Sun, and Zhuang Miao. 2018. A set-based genetic algorithm for interval many-objective optimization problems. *IEEE Transactions on Evolutionary Computation* 22, 1 (2018), 47–60.

[7] Sandhya Harikumar and Raji Ramachandran. 2015. Hybridized fragmentation of very large databases using clustering. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems*. 1–5.

[8] Jiajia Huang, Min Peng, Hua Wang, Jinli Cao, Wang Gao, and Xiuzhen Zhang. 2017. A probabilistic method for emerging topic tracking in microblog stream. *World Wide Web* 20, 2 (2017), 325–350.

[9] Ravi Jhawar, Vincenzo Piuri, and Pierangela Samarati. 2012. Supporting security requirements for resource management in cloud computing. In *2012 IEEE International Conference on Computational Science and Engineering*. IEEE, 170–177.

[10] Alhassan Khedr, Glenn Gulak, and Vinod Vaikuntanathan. 2016. SHIELD: scalable homomorphic implementation of encrypted data-classifiers. *IEEE Trans. Comput.* 65, 9 (2016), 2848–2858.

[11] Jens Köhler, Konrad Jünemann, and Hannes Hartenstein. 2015. Confidential database-as-a-service approaches: taxonomy and survey. *Journal of Cloud Computing* 4, 1 (2015), 1.

[12] Lichun Li, Rongxing Lu, Kim-Kwang Raymond Choo, Anwitaman Datta, and Jun Shao. 2016. Privacy-preserving-outsourced association rule mining on vertically partitioned databases. *IEEE Transactions on Information Forensics and Security* 11, 8 (2016), 1847–1861.

[13] Meikang Qiu, Zhong Ming, Jiayin Li, Keke Gai, and Ziliang Zong. 2015. Phase-change memory optimization for green cloud with genetic algorithm. *IEEE Trans. Comput.* 64, 12 (2015), 3528–3540.

[14] Jiangang Shu, Xiaohua Jia, Kan Yang, and Hua Wang. 2018. Privacy-preserving task recommendation services for crowdsourcing. *IEEE Transactions on Services Computing* (2018).

[15] Suk-Kyu Song and Narasimhaiah Gorla. 2000. A genetic algorithm for vertical fragmentation and access path selection. *Comput. J.* 43, 1 (2000), 81–93.

[16] R Srinivas, KA Sireesha, and Shaik Vahida. 2017. Preserving Privacy in Vertically Partitioned Distributed Data Using Hierarchical and Ring Models. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*. Springer, 585–596.

[17] Xiaoxun Sun, Hua Wang, Jiuyong Li, and Yanchun Zhang. 2012. Satisfying privacy requirements before data anonymization. *Comput. J.* 55, 4 (2012), 422–437.

[18] Noorul Hussain UbaidurRahman, Chithralekha Balamurugan, and Rajapandian Mariappan. 2015. A novel DNA computing based encryption and decryption algorithm. *Procedia Computer Science* 46 (2015), 463–475.

[19] Hua Wang, Zonghua Zhang, and Tarek Taleb. 2018. Special issue on security and privacy of IoT. *World Wide Web* 21, 1 (2018), 1–6.

[20] Xiaofeng Xu, Li Xiong, and Jinfei Liu. 2015. Database fragmentation with confidentiality constraints: A graph search approach. In *2015 ACM Conference on Data and Application Security and Privacy*. 263–270.

[21] Jiangshan Yu, Guilin Wang, Yi Mu, and Wei Gao. 2014. An efficient generic framework for three-factor authentication with provably secure instantiation. *IEEE transactions on information forensics and security* 9, 12 (2014), 2302–2313.

[22] Yong Yu, Man Ho Au, Giuseppe Ateniese, Xinyi Huang, Willy Susilo, Yuanshun Dai, and Geyong Min. 2017. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. *IEEE Transactions on Information Forensics and Security* 12, 4 (2017), 767–778.

[23] Ji Zhang, Xiaohui Tao, and Hua Wang. 2014. Outlier detection from large distributed databases. *World Wide Web* 17, 4 (2014), 539–568.