Soft Computing (2023) 27:13911–13930
https://doi.org/10.1007/s00500-023-08452-x

OPTIMIZATION

A new approach based on greedy minimizing algorithm for solving data allocation problem

Mostafa Mahi[1] · Omer Kaan Baykan[2] · Halife Kodaz[2]

Accepted: 2 May 2023 / Published online: 23 May 2023

Abstract
Distributed database functionality depends on sites responsible for the allocation of fragments. The aims of the data allocation problem (DAP) are to achieve the minimum execution time and ensure the lowest transaction cost of queries. The solution for this NP-hard problem based on numerical methods is computationally expensive. Despite the success of such heuristic algorithms as GA and PSO in solving DAP, the initial control parameters tuning, the relatively high convergence speed, and hard adaptations to the problem are the most important disadvantages of these methods. This paper presents a simple well-formed greedy algorithm to optimize the total transmission cost of each site-fragment dependency and each inner-fragment dependency. To evaluate the effect of the proposed method, more than 20 standard DAP problems were used. Experimental results showed that the proposed approach had better quality in terms of execution time and total cost.

Keywords Data allocation problem · Greedy algorithm · Particle swarm optimization · Distributed databases system · Site-fragment dependency

1 Introduction

Recently, one of the most attractive applications has been the development of a distributed database called data allocation problem (DAP). The aim of DAP is to determine the fragments placement in various sites to reduce the total transaction cost when the query is taken from one site to another one. The optimization algorithms of performance analysis with special constraints are used for DAP with the standard test problem (Tosun 2014a; Tosun et al. 2013a).

✉ Halife Kodaz
hkodaz@ktun.edu.tr

Mostafa Mahi
mostafamahi@gmail.com

Omer Kaan Baykan
okbaykan@ktun.edu.tr

[1] Computer Engineering and Information Technology Department, Payame Noor University, PO Box 19395-3697, Tehran, Iran

[2] Computer Engineering Department, Engineering and Natural Sciences Faculty, Konya Technical University, Konya, Turkey

The problems of data allocation for sites are very difficult. The data for the locations of the fragments can be changed. In this situation, the data organization becomes more important. For instance, some items such as parallel query executions, network load, and the server load balancing need to be managed. DAP is an NP-hard problem, without considering the problems mentioned above. DAP can be solved by two types of algorithms: dynamic and static. The static algorithms are implemented based on data allocation on static transaction execution patterns in the target environment. These patterns are changed in the dynamic algorithm (Tosun et al. 2013a; Gu et al. 2006; Mashwani and Salhi 2012). DAP has been solved by several algorithms such as Genetic Algorithm (Tosun et al. 2013a; Mashwani and Salhi 2012), Ant colony optimization (Tosun 2014b; Adl and Rankoohi 2009), Particle Swarm Optimization (Mahi et al. 2015, 2018), and Metaheuristic methods. In this part of the paper, we have reviewed some studies on the DAP solution. Peng et al. (2022) propose an allocation scheme for the storage of data in a collaborative edge-cloud environment, with a focus on enhanced data privacy. Specifically, they first divide the datasets by fields

to eliminate as much as possible the correlation between the leaked data.

Anita Brigit Mathew proposed a heuristic algorithm based on separating a database graph among nodes by defining all information on the same or adjacent nodes (Mathew 2018). This heuristic algorithm includes the best-fit decrease with Ant Colony Optimization, which refers to the data allocation in the distributed architectures of the NoSQL database graph (Mathew 2018). An effective data allocation method, which contemplates static and dynamic specifications of data centers to make more effectual datacenter resizing, was proposed by Chen et al. (2018); they used a heuristic algorithm for analyzing the current traffic in the network of data centers through first transmitting the data allocation problem into a chunk distribution tree (CDT). An improved heuristic method based on division and allocation has been proposed by Amer et al. (2012, 2017) all of the mentioned methods are combined into a single, efficient one, which has an effectual solution for Distributed Database Systems (DDBS). Nashat et al. (2018) proposed a method based on a complete taxonomy of the accessible division and allocation in the distributed database schema. Data division in the DDBS has been surveyed by Asma et al. (2017) An improved data allocation through data migration algorithm on task level (TODMA) has been presented by Du et al. (2017); Mayne and Satav (2017). Simulated annealing (SA) is done by Sen et al. to solve DAP (Sen et al. 2016). Radio Frequency Identification (RFID) tag oriented DAP as a nonlinear knapsack problem has been modeled by Wang et al. The heuristic Quadratic Assignment Problem (QAP) was designed and implemented for DAP by Tusun et al. (2013a) They proposed a fast and scalable hybrid genetic multi-start tabu search algorithm that outperformed the other well-known heuristics in terms of the execution time and solution quality (Tosun 2014a; Tosun et al. 2013a). Tosun et al. have presented a set of SA, GA, and fast ACO to solve DAP (Tosun 2014a). Nasser et al. also proposed an innovative hybrid method. Differential Evolution and Variable Neighborhood Search (DEVNS) have also been offered for solving DAP in distributed database systems (Lotfi 2019).

ACO-DAP model based on ACO and local search has been presented by Adl and Rankoohi (2009). In this approach, overcoming on RAPs has been targeted. Genetic algorithms were considered in their method and the simulation results demonstrated that its performance was good. Ulus and Uysal also presented a new dynamic DDBS called threshold algorithm (Ulus and Uysal 2003). In this approach, data reallocation has been done by changing the data access pattern. The obtained results were compared with the genetic algorithm (Tosun 2014a), Tabu search (Tosun 2014a), ant colony (Tosun 2014a), and simulated annealing (Tosun et al. 2013b) in regard to solving 20 problems with various dimensions. The execution time and the total cost were important factors. The proposed algorithm had suitable and comparable results in time; it could be inferred that Greedy-DAP execution time, in comparison with other algorithms, could be regarded as the best. In our work, we want to solve DAP through Greedy-DAP utilization and adaptation. The execution time and fragment allocation quality are investigated experimentally by Greedy-DAP. The simulation results reveal that the Greedy-DAP's (Mahi et al. 2018) execution time together with cost could have a suitable performance in comparison with other algorithms.

Three goals for data allocation are presented, Cao (Cao 2022), minimizing the number of active servers, minimizing the average number of partitions per data, and balancing servers' workload. Li et al. (2022) are proposed demonstrates that the conventional "graph data allocation = graph partitioning" assumption is not true, and the memory access patterns of graph algorithms should also be taken into account when partitioning graph data for communication minimization. Thalij (2022), a novel high-performance data allocation approach, is designed using Chicken Swarm Optimization (CSO) algorithm. Then the CSO algorithm optimally chooses the sites for each of the data fragments without creating much overhead and data route diversions. Then, the CSO algorithm optimally chooses the sites for each of the data fragments without creating much overhead and data route diversions. Then, the CSO algorithm optimally chooses the sites for each of the data fragments without creating much overhead and data route diversions. This scenario is formulated using an optimization problem called the data allocation problem (DAP). In this paper, in addition to the fact that algorithms based on randomness are not used, we directly find the best solution to the problem in each step and refer to the next steps, and in the new step, we find the best solution and move to the next step. Of course, at each stage after the best solution for resource allocation on the site is obtained, in the matrix of costs, the column in which the part is selected, we put all the columns of that matrix as a very large number. Unfortunately, this problem is that in the next steps, maybe the best answer is again in that column, where we lose this amount. However, according to the obtained results, we have the minimum answer compared to the previous algorithms. The meaning of the best answer, according to the resource allocation matrix on the sites, is the minimum value found in the entire cost matrix. Which is done by replacing the source on the site with the lowest transaction cost. Finally, after replacing the source on the site, we put a very large number matrix in the columns of that column, so that in the next steps, the same minimum will not be found again from that column.

Table 1 Description of notations (Adl and Rankoohi 2009)

Symbol	Description
n	The number of sites
m	The number of fragments
i	The index of sites
j	The index of fragments
S_i	The ith site
$SiteCap_i$	The storage capacity of site S_i
$UC_{n\times n}$	The matrix denoting the cost of unit data transmission between each two sites.
uc_{i1i2}	The cost of sending a unit data item from site S_{i1} to the site S_{i2}.
f_j	The jth fragment
$fragSize_j$	The size of fragment
L	The number of considered transactions
t_k	The kth transaction
$FREQ_{n\times l}$	The matrix denoting the execution frequency of each transaction in each site
$freq_{ik}$	The execution frequency of transaction t_k in $sites_i$
$TRFR_{l\times m}$	The matrix denoting the direct transaction − fragment dependency
$trfr_{kj}$	The volume of data items of fragment f_j that must be sent
	from site containing f_j to the site executing transaction t_k, for each execution of t_k
$Q_{l\times m\times m}$	The matrix denoting the indirect transaction − fragment dependency
q_{kj1j2}	The volume of data items that must be sent from site containing fragment f j1 to the site storing $f_{j2,}$ for each execution of transactiont_k
Ψ	The m element vector which denotes an allocation scheme
Ψ_j	The site to which fragment t_k is assigned in the allocation scheme
$COST(\Psi)$	The cost of data transmission in an allocation scheme Ψ
$COST1(\Psi)$	The cost of data transmission in an allocation scheme Ψ resulting from direct transaction − fragment dependencies
$COST2(\Psi)$	The cost of data transmission in an allocation scheme Ψ resulting from indirect transaction fragment dependencies
$STFR_{n\times m}$	The matrix denoting the site − fragment dependency
$stfr_{ij}$	The volume of data items from fragment f_j time (according to the site − fragment dependency) which are accessed by site s_i in unit
$PARTIALCOST1_{nxm}$	The matrix denoting the $COST1(\Psi)$ incurred by allocating each fragment to each site
$partialcost1_{ij}$	The cost incurred by f j allocated to site $s_i$as a result of direct transaction fragment dependency
$QFR_{l\times m\times m}$	The matrix denoting the indirect transaction fragment dependency taking the execution frequencies of
	The transactions into account
qfr_{kj1j2}	The volume of data needed to be sent from site storing fragment $f_{j1}$to the site having fragment f_{j2} in unit time taking into account the transaction frequency of t_k
$FRDEP_{m\times m}$	The matrix denoting the inter fragment dependency
$frdep_{j1j2}$	The volume of data items needed to be sent from site having fragment f_{j1} to the site having fragment f_{j2}
	dependency in unit time due to the indirect transaction fragment
ParticleNumber	The number of Particle

Table 1 (continued)

Symbol	Description
V	The velocity of Particle
X	The position of Particle
TotalCap_i	The current capacity of site S_i
Iteration Number	The number of iteration
W	Inertia weight
S	The count number of plus signs
X_s	The mean of the binomial distribution
σ_s	The standard deviation of the binomial distribution
Z	Test statistic
H_0	There is no significant difference between the two algorithms
H_1	There is a significant difference between the two algorithms
Max	Maximum element in the Cost matrix

Section 1 serves as the introduction and the rest of the paper is structured as follows; Sect. 2 contains materials and methods of the background information for the greedy algorithm and introduces the proposed method (Greedy-DAP). Comparisons and the experimental results are presented in Sect. 3. Finally, Sect. 4 will conclude the paper.

2 Materials and methods

A greedy algorithm is a simple and intuitive algorithm used in optimization problems. It has a function that calculates the optimal choice made at each step along with finding the overall optimal method to solve the entire problem. Two examples of the problems which can be solved successfully using greedy algorithms including Huffman encoding and Dijkstra's algorithm; these are used to compress data and find the shortest path through a graph, respectively. The greedy algorithm operates in a way that takes all of the data to a certain problem and sets a rule for the elements to add solution at each step of the algorithm (Astrachan, et al. 2002). Kadam and Kim (2022) show that it is NP-Complete and propose a greedy algorithm to solve it. Table 1 refers to the notations.

2.1 Data allocation problem

The purpose of DAP is to find the location of fragments in the best sites, to alleviate the total cost of a transaction when the query is taken from one site to another one (Adl and Rankoohi 2009; Mahi et al. 2018; Mamaghani, et al. 2010). Figure 1 shows the dependencies among sites, fragments, and transactions (Adl and Rankoohi 2009). For example, to get a query from S_1 to S_2 for obtaining the fragment j, the transaction k is necessary. Transaction access to the website is done the through site Fragment Frequency (FREQ) matrix, which included frequency values among sites transactions. Transactions to fragmentations achievements are done through the Transactions to Fragmentations (TRFR) matrix. Evaluating the amount of data transactions for fragments dependency is done through the TRFR matrix, which has some parameters. The Q matrix refers to the data between two fragments of one transaction. The size of each fragment is at interval $\left[\frac{c}{10}, \frac{20*c}{10}\right]$, which is chosen randomly. In this range, c is a value at the interval [*101,000*] (Adl and Rankoohi 2009).

The size of each fragment is calculated by Eq. (1) (Adl and Rankoohi 2009).

$$\left(\sum_{\mathrm{i}=1}^{\mathrm{n}} p_i = \mathrm{m}\right), rf_i = m - \sum_{\mathrm{q}=1}^{\mathrm{i}} p_q \tag{1}$$

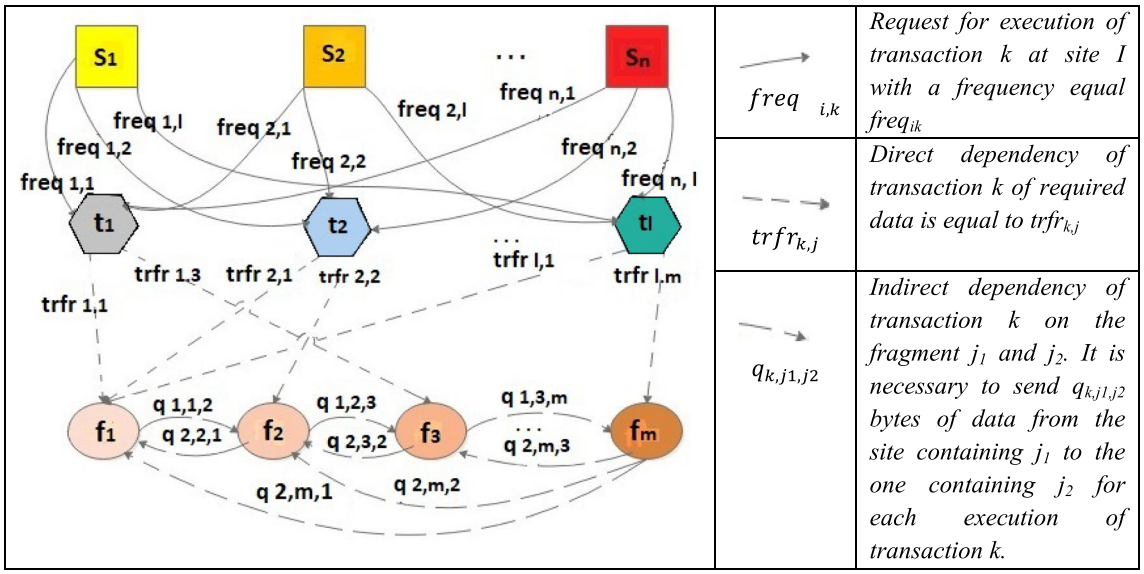

Fig. 1 The dependences among sites, transactions and fragments (Adl and Rankoohi 2009)

Also, the site capacity site is calculated by Eq. (2) (Adl and Rankoohi 2009).

$$siteCap_i = p_i * max_{1 \le j \le m}(fragSize_j) \quad (2)$$

The site capacity should not be more than $\text{fragSize}_j * x_{ij}$ during the fragments replacement on the site, as calculated by Eq. (3) (Adl and Rankoohi 2009).

$$\sum_{j=1}^{m} \text{fragSize}_j * x_{ij} \le \text{siteCap}_i \text{i} = 1, 2, \ldots, n \quad (3)$$

COST1 is calculated from the allocation of the fragments on sites according to Eq. (4)) (Adl and Rankoohi 2009). According to fragment size and site capacity, fragments are allocated to sites and the vector is created.

$$\text{partialcost1}_{ij} = \sum_{q=1}^{n} \text{uc}_{iq} * \text{stfr}_{qj} \quad (4)$$

Vector (Eq. (5)) (Adl and Rankoohi 2009) is related to the COST1 calculation.

$$\text{COST1}(\psi) = \sum_{j=1}^{m} \text{partialcost1}_{\psi j^j} \quad (5)$$

COST2, the parameter calculated as the query from the site j1 to j2, has been taken. The COST parameter is calculated through the matrix q. The matrix is calculated through multiplying the sum of the column with the element of the *freg* matrix by the matrix (Eq. (6)) (Adl and Rankoohi 2009).

$$\text{qfr}_{kj1j2} = q_{kj1j2} * \sum_{r=1}^{n} \text{freq}_{kr} \quad (6)$$

FRDEP matrix is calculated based on the accumulation of the transaction cost between fragments (Eq. (7)) (Adl and Rankoohi 2009).

$$frdep_{j1j2} = \sum_{k=1}^{l} qfr_{kj1j2} \quad (7)$$

COST2 is achieved through the FRDEP matrix and vector, as shown by Eq. (8) (Adl and Rankoohi 2009), through multiplying.

$$\text{COST2}(\psi) = \sum_{j1=1}^{m} \sum_{j2=1}^{m} \text{frdep}_{j1j2} * \text{uc}_{\psi_{j1}\psi_{j2}} \quad (8)$$

Finally, the sum of COST1 and COST2 is related to the COST according to the vector produced as Eq. (9) (Adl and Rankoohi 2009). The vector which is created to allocate fragments on sites to get the query for the algorithm in this paper is the best. The mentioned aim will be pursued by our new method, as discussed in the next section.

$$COST(\psi) = COST1(\psi) + COST2(\psi) \quad (9)$$

2.2 The proposed method (Greedy-DAP)

DAP solution based on greedy algorithm utilization and adaptation is the aim of this study. The method presented can be called Greedy-DAP. It is an innovative approach to solve DAP. Due to the fewer control parameters of the greedy algorithm, some parameters such as speed convergence specifications, low consuming time, and robustness against the solution space of the optimization problems are rarely used to solve the optimization problems with the same characteristics by the vector p (Deng et al. 2012; Bai

Table 2 Cost matrix representation

		1	2	...	J	...	m
	1						
	2						
	⋮						
Cost Matrix =	i	...	...	...	Min		
	⋮						
	n						

n x m

2010). To achieve this goal, the cost must be low; then, to reduce the cost of transactions, fragments must be implemented in the sites. *Vectorp* is used to compute the cost of fragments replacement in DAP. In the Greedy-DAP, the size of the vector p is $1 * m$ and its structure is as follows:

	1	2	…	j	…	m	Fragment number
Vector p:	2	3	…	i	…	n	Site number of the fragment

Vectorp is an array, indicating that the fragment is located on the site. For example, fragment 1 is located on-site 2, fragment 2 is located on-site 3, etc. *Vectorp* shows which fragment will be placed on which site. Fragments placement on the best site to alleviate the total transaction cost is our aim. This greedy algorithm, which considers the total cost, is used to evaluate the process of fragment allocation to the sites. Fitness calculation is done by cost function in the greedy algorithm, as described in Sect. 2.1. Determination of the cost values of the vectors is done by Eq. (10), and the Max number is equal to the value of the maximum element in the Cost matrix.

$$\begin{aligned} Cost[i,j] &= ((partial\,Cost[i,k] * (frdep[k,j]) \\ &\quad * (Uc[k,j])), Max\,number \\ &= maximum\,Cost[i,j] \end{aligned} \tag{10}$$

Site capacity is one of the parameters in Greedy-DAP; so, a counter is determined for each site to check the capacity of the site. Cost matrix has n rows and m columns, where each row and column represent a site and fragment, respectively. Firstly, the minimum value of the cost matrix is found in the $Cost[i,j]$. If the fragment size j is greater than the capacity site i, we should find a second minimum element in the Cost matrix. We should continue finding the $k-th$ minimum until the fragment size j is less than site capacity i and update the site capacity by reducing the site capacity from fragment size. Secondly, if the above

Table 3 Updated cost matrix

		1	2	...	j	...	m
	1				Max		
	2				Max		
	⋮				Max		
Cost =	i	...	...	...	Max		
	⋮				Max		
	n				Max		

n x m

Fig. 2 Pseudo-code of the Greedy-DAP

```
- Initialization (number of fragment(m) and number of site(n), iteration number, length of the P, site
capacities )
Generate data set standard:
-Create Cost
      -Determination of the allocation of fragment on sites of the Cost
                  find min cost from Cost
            If site capacities are overflow
                Fragment does not replace on that site.
            Else allocate fragments to the sites And Update Site Capacity j
                The values of column j are replaced by value.
       - find again minimum from Cost
      - Specify the vector to allocate fragments to the sites
     -updating the location of Cost
```

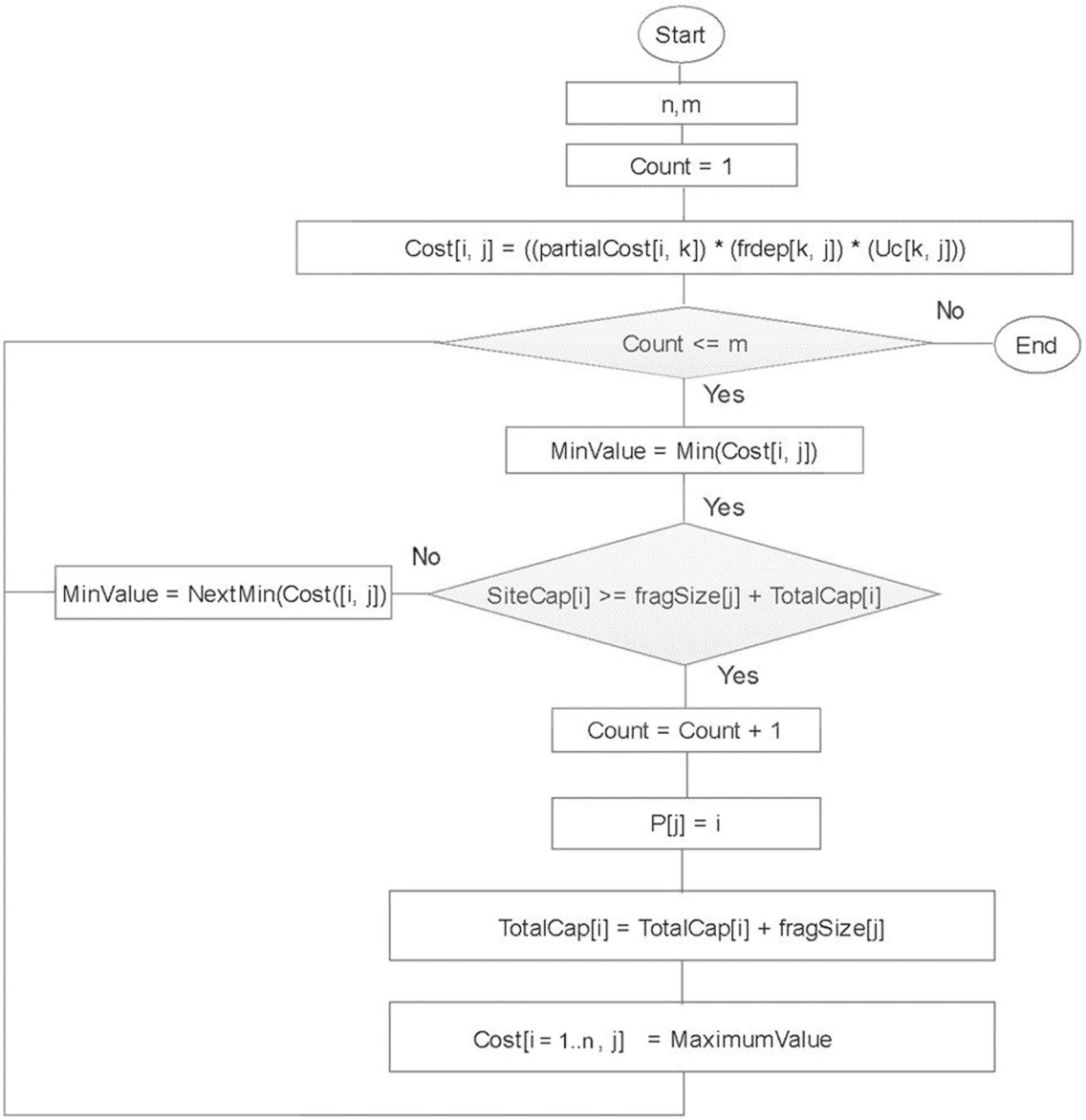

Fig. 3 The scripting chart of the Greedy-DAP

Table 4 Input parameters for PSO-DAP (Mahi et al. 2018)

Parameter description	Parameter Name	Value
Approximation of the average fragment size	C	10
Unit transmission cost between two neighbor sites	UCN	[0–1]
Number of transactions	L	20
Probability of a transaction being requested at a site	RPT	0.7
Probability of a fragment being accessed by a transaction	APF	0.4
Probability of a transaction necessitates data transaction between two sites (Other than theoriginating site)	APFS	0.025
Number of particle	P	30
Learning factors	c_1, c_2	2
Number of iteration	K	500
Inertia weight	W	0,5
Maximum velocity	V_{max}	N
Generate random number	$rand_1$, $rand_2$	[0–1]

Table 5 Generate cost and execution time for increasing site numbers (n) and fragment number is fixed by 48

n	Cost $\times 10^9$	Time (s)	n	Cost $\times 10^9$	Time (s)
3	0.002	0.902	26	0.192	8.093
4	0.004	1.164	27	0.223	8.470
5	0.005	1.488	28	0.232	9.236
6	0.009	1.909	29	0.247	9.330
7	0.015	2.275	30	0.236	10.254
8	0.025	2.450	31	0.262	10.297
9	0.019	2.750	32	0.300	10.175
10	0.030	3.166	33	0.324	10.692
11	0.043	3.498	34	0.301	11.845
12	0.036	4.069	35	0.330	12.273
13	0.047	4.158	36	0.384	11.525
14	0.056	4.264	37	0.347	11.640
15	0.042	4.662	38	0.427	13.358
16	0.075	5.216	39	0.414	13.810
17	0.082	5.464	40	0.468	14.476
18	0.099	5.567	41	0.527	13.810
19	0.110	6.143	42	0.544	13.892
20	0.085	6.073	43	0.522	14.702
21	0.104	7.674	44	0.565	16.328
22	0.142	6.941	45	0.639	15.306
23	0.151	7.816	46	0.597	16.405
24	0.167	8.070	47	0.619	17.472
25	0.128	8.092	48	0.671	15.557

condition is true in the $Cost[i,j]$, then we can put j into index i in the vector p. After that, the column j from the Cost matrix is filled with the Max number. The Cost matrix is given in Table 2.

Then we will find the next minimum of the matrix; we will continue this process until all elements in the matrix are filled up to maximum value. Replace the values of the column j in a maximal matrix. The updated Cost matrix is given in Table 3.

Greedy-DAP pseudo-code and consequently its scripting chart are shown in Figs. 2 and 3, respectively.

Table 6 Cost comparison of methods for increasing site numbers (n) and fragment number is fixed by 48 (cost values are column $\times 10^9$)

n	PSO-DAP			Greedy-DAP (Proposed Method)	n	PSO-DAP			Greedy-DAP (Proposed Method)
	Average	Minimum	Standard deviation			Average	Minimum	Standard deviation	
3	0.0004	**0.0004**	0.000	**0.0004**	26	0.183	0.178	0.008	**0.173**
4	0.002	**0.002**	0.000	**0.002**	27	0.216	0.211	0.020	**0.201**
5	0.003	**0.003**	0.000	**0.003**	28	0.225	0.221	0.025	**0.209**
6	0.004	**0.004**	0.000	**0.004**	29	0.244	0.235	0.016	**0.206**
7	0.009	0.009	0.000	**0.008**	30	0.226	0.216	0.022	**0.201**
8	0.022	**0.022**	0.000	0.023	31	0.255	0.253	0.019	**0.235**
9	0.017	0.016	0.004	**0.013**	32	0.301	0.294	0.012	**0.260**
10	0.025	0.023	0.005	**0.020**	33	0.308	0.306	0.030	**0.272**
11	0.037	0.035	0.001	**0.032**	34	0.290	0.277	0.044	**0.271**
12	0.031	**0.030**	0.001	0.033	35	0.319	0.314	0.028	**0.281**
13	0.035	**0.033**	0.000	0.036	36	0.372	0.370	0.048	**0.330**
14	0.042	**0.042**	0.001	0.044	37	0.336	0.333	0.051	**0.321**
15	0.035	0.033	0.001	**0.026**	38	0.419	0.415	0.078	**0.397**
16	0.071	0.061	0.002	**0.059**	39	0.388	0.381	0.078	**0.341**
17	0.074	0.070	0.002	**0.066**	40	0.436	0.434	0.113	**0.402**
18	0.086	0.084	0.003	**0.082**	41	0.520	0.512	0.069	**0.477**
19	0.101	0.100	0.004	**0.088**	42	0.535	0.533	0.077	**0.480**
20	0.071	0.064	0.002	**0.064**	43	0.511	0.509	0.116	**0.488**
21	0.094	0.090	0.005	**0.088**	44	0.541	0.526	0.111	**0.472**
22	0.114	0.113	0.002	**0.109**	45	0.616	0.614	0.109	**0.585**
23	0.137	0.134	0.006	**0.121**	46	0.579	0.561	0.163	**0.502**
24	0.158	0.156	0.017	**0.138**	47	0.621	0.616	0.166	**0.565**
25	0.125	0.123	0.005	**0.119**	48	0.641	0.634	0.109	**0.590**

3 Experimental results

The original data set has been produced based on the cost formulation in Sect. 2.2, in order to compare DAP with other algorithms. The proposed algorithm was evaluated using three cases of comparisons in various fragments and site sizes. In the first case, the number of fragments was equal, but the number of sites was variable. In the second case, the number of sites was equal, but the number of fragments was different. The third case contained the same size of fragments and sites. It is worth noting that the proposed algorithm is a deterministic one; therefore, the results of the iterations of the algorithm were the same. However, due to using a random dataset, the cost and execution time of other algorithms are different. In order to get the minimum number of results, the program was tested in 20 iterations of executions in PSO-DAP, whereas in the case of Greedy-DAP, due to obtaining the same results in all executions, only one iteration of execution was done. PSO-DAP iteration was done in 500. Computer parameters for comparing PSO-DAP and Greedy-DAP based on an algorithm presented for DAP can be listed as follows: CPU: 1.6 GHZ, memory: 4 GB, Windows 7, and C# programming language.

3.1 State 1

The first state refers to increasing the site size from 3 to 48 and fixing the number of fragments in 48. The values of the algorithms in paper (Tosun et al. 2013a) have just been shown in the figure. The results of the cost and execution time for PSO-DAP and Greedy-DAP are shown in Tables 6 and 7, respectively. In order to compare the obtained results of the proposed algorithm with a figure in paper (Tosun et al. 2013a), we have mapped the proposed algorithm results to those charts. Considering that other algorithms used the same approach; forty-six different sites numbers were taken from the range of 3 to 48 and the number of fragments were fixed by 48. Input parameters in Table 4, as shown in bold, refer to PSO-DAP and others are common in both PSO-DAP and Greedy-DAP.

Table 7 Execution time (s) comparison of methods for increasing site numbers (n) and fragment number is fixed by 48

n	PSO-DAP Minimum	Greedy-DAP (Proposed method)	n	PSO-DAP Minimum	Greedy-DAP (Proposed method)
3	15.466	0.366	26	18.158	7.956
4	16.021	0.069	27	16.910	8.767
5	15.538	0.133	28	17.082	9.438
6	15.007	0.210	29	16.942	10.296
7	15.694	0.299	30	16.630	10.967
8	15.772	0.455	31	16.973	11.872
9	15.787	0.608	32	16.770	12.683
10	15.818	0.718	33	16.942	13.603
11	16.006	0.905	34	17.176	15.803
12	15.928	1.061	35	17.035	16.068
13	15.694	1.232	36	17.082	16.520
14	15.694	1.498	37	17.410	17.456
15	15.803	1.685	38	17.098	19.578
16	15.678	1.950	39	19.016	20.608
17	17.690	2.200	40	16.957	23.026
18	17.893	2.761	41	16.957	24.508
19	18.174	3.635	42	17.534	25.802
20	17.612	3.635	43	17.300	27.035
21	18.580	4.664	44	18.143	28.548
22	19.781	5.320	45	18.923	30.420
23	19.687	5.897	46	17.160	32.105
24	16.739	6.646	47	16.879	33.680
25	16.926	7.348	48	16.754	33.758

Samples of the DAP cost and execution time values are shown in Table 5. The obtained results of the Greedy-DAP algorithm were compared with the other methods in the literature, namely Ant γ Algorithm (Adl and Rankoohi 2009), Ant β Algorithm (Adl and Rankoohi 2009), Ant α Algorithm (Adl and Rankoohi 2009), Evolutionary (Adl and Rankoohi 2009) and PSO-DAP minimum.

Tables 6 and 7 contain the cost and execution time values, respectively; the best results are shown in bold.

A comparison of the obtained results demonstrated that Greedy-DAP had less cost and was a less time-consuming process in comparison with PSO-DAP. DAP samples were created randomly and Greedy-DAP and PSO-DAP used them to solve problems in this paper. Datasets were created randomly according to the formula described in Sect. 2.2 for the purpose of comparison with the previous algorithm's datasets. The results demonstrated that, among 46 obtained results, 39 were the best in terms of cost, and they were almost similar to our compared algorithm's results in Tables 6 and 8, thereby showing that the proposed algorithm was statistically different from other methods. Consequently, PSO-DAP would consume less time than other algorithms, as shown in Table 7 (Mahi et al. 2018).

Here, to show the accuracy of our method, we used the sing test (Mann 2013; Lurie et al. 2011). This test sign was done for three states. There was no significant difference between the two algorithms as the H0 hypothesis and there was a significant difference between the two algorithms as the H1 hypothesis. All calculations were implemented at a significance level of five percent. Table 8 contains the sign test results. The H1 hypotheses were accepted because the computed Z values were outside the range in all tests. The proposed algorithm and other alternatives related to comparison results are shown in the last two rows of Table 8.

In Figs. 4, 5, 6, 7, we plot the Greedy-DAP cost and the time results from comparisons with other approaches such as Ant γ Algorithm (Adl and Rankoohi 2009), Ant β Algorithm (Adl and Rankoohi 2009), Ant α Algorithm (Adl and Rankoohi 2009), Evolutionary (Adl and Rankoohi 2009) and PSO-DAP minimum (Mahi et al. 2018).

3.2 State 2

The second state of the algorithm refers to increasing the fragment size from 20 up to 50 by step 1 and fixing the number of sites in 20 (Table 9). The obtained results of the

Table 8 Statistical comparison of the methods using sign test for increasing site numbers (n) with fixed fragment number (48)

n	Greedy-DAP (Proposed Method)	PSO-DAP minimum	Sign	n	Greedy-DAP (Proposed Method)	PSO-DAP minimum	Sign
3	441,872	410,603	+	26	172,615,295	177,702,535	–
4	1,664,678	1,244,572	+	27	200,975,127	211,071,504	–
5	2,580,880	2,591,430	–	28	208,627,304	221,408,655	–
6	3,918,779	3,898,362	+	29	205,512,776	234,776,499	–
7	8,475,686	8,892,369	–	30	201,362,336	216,136,938	–
8	22,671,850	21,702,160	+	31	234,527,270	253,376,973	–
9	12,915,310	16,140,468	–	32	260,187,978	293,588,991	–
10	20,473,330	23,256,179	–	33	272,106,382	306,261,617	–
11	32,423,834	35,198,738	–	34	271,027,460	277,055,891	–
12	33,156,448	30,360,039	+	35	280,603,467	314,138,626	–
13	35,662,487	33,175,773	+	36	330,381,584	370,207,013	–
14	44,037,891	41,718,648	+	37	320,758,035	333,278,681	–
15	26,337,352	32,883,834	–	38	397,051,238	414,989,215	–
16	59,321,311	60,711,667	–	39	341,185,743	381,319,377	–
17	66,032,543	70,417,580	–	40	401,932,655	433,510,525	–
18	82,477,752	84,454,729	–	41	477,285,889	512,475,049	–
19	88,167,561	99,960,603	–	42	479,891,345	532,748,715	–
20	64,205,904	64,473,334	–	43	487,529,241	508,885,582	–
21	87,930,403	89,750,233	–	44	471,533,110	526,005,587	–
22	109,131,359	113,228,406	–	45	585,214,613	614,206,988	–
23	121,130,352	134,426,817	–	46	501,554,620	560,510,603	–
24	138,049,507	156,493,230	–	47	565,195,696	615,874,059	–
25	118,674,870	122,929,074	–	48	590,016,599	633,864,873	–

Statistical Notations	Greedy-DAP vs PSO-DAP
	7
	22.5
	3.354
	− 4.621
H_0	Reject
H_1	**Accept**

algorithm in paper (Adl and Rankoohi 2009) have not been shown in the table. The results of the cost and execution time of PSO-DAP and Greedy-DAP are shown in Tables 10 and 11. To compare the results of the paper (Adl and Rankoohi 2009), we mapped our obtained results to those charts, as shown in Figs. 6 and 7. These figures show that in comparison with other algorithms, the proposed one has the lowest cost and time. Greedy-DAP and the achieved results have been compared with other methods, such as Ant γ Algorithm (Adl and Rankoohi 2009), Ant β Algorithm (Adl and Rankoohi 2009), Ant α Algorithm (Adl and Rankoohi 2009), Evolutionary (Adl and Rankoohi 2009) and PSO-DAP (Mahi et al. 2015).

The DAP samples size increment in terms of the cost values and execution time is shown in Table 9.

Table 10 shows the results related to cost from three algorithms and also the standard deviation for the proposed algorithm. Table 11 shows the execution times of different methods.

The statistical comparison of the proposed method and the other method is given in the last six rows of Table 12 for increasing the fragment numbers (m) and the fixed site number, showing the acceptable results.

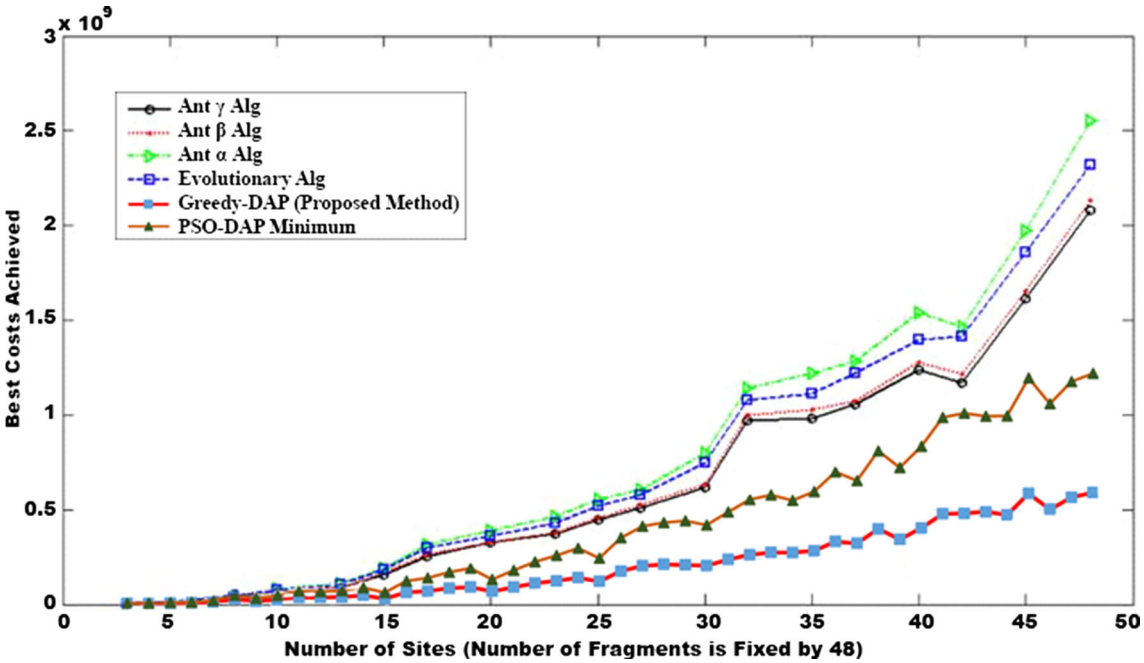

Fig. 4 Evaluating the Results achieved by the algorithms in a state 1 comparison for cost (Adl and Rankoohi 2009)

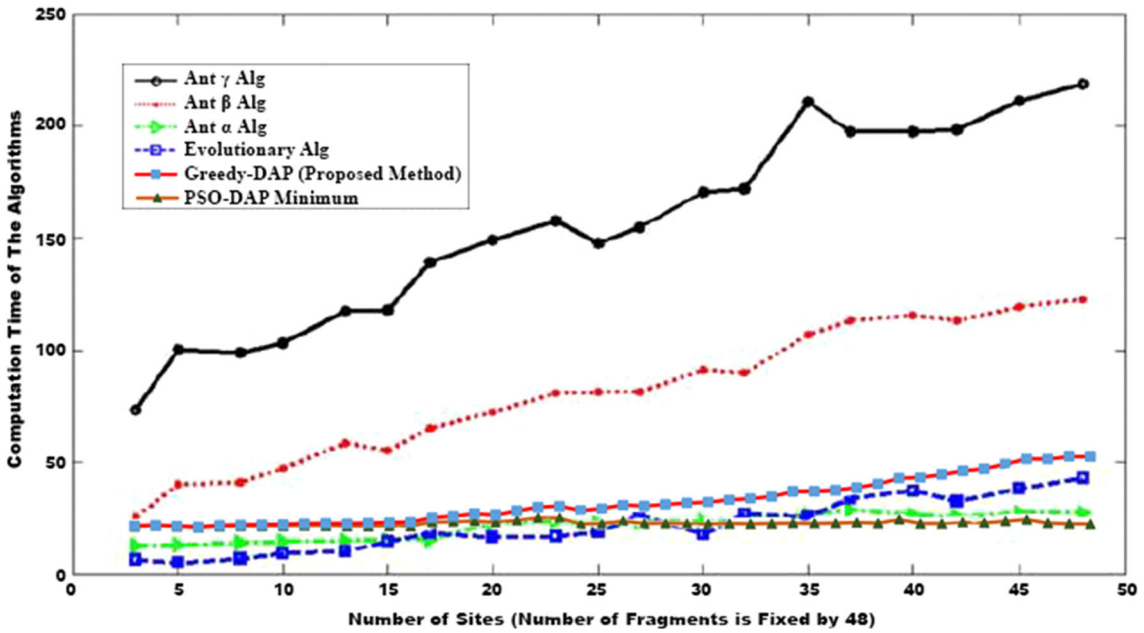

Fig. 5 Evaluating the Computation time of the algorithms in a state 1 comparison for time (Adl and Rankoohi 2009)

3.3 State 3

The number of fragments and sites are equal so our proposed algorithm, as implemented on an equal number of fragments and sites. DAP samples size increment cost values and execution time generations are shown in Table 13.

Tables 14 and 15 present the comparison covering the cost and execution time values, respectively. The best results are shown in bold. Table 14 shows the cost of the results, which is much better than the results of the previous algorithms. In Table 15, the calculation time of resource allocation on the sites is shown that the more the number of sites with their resources, the more time it takes to get them.

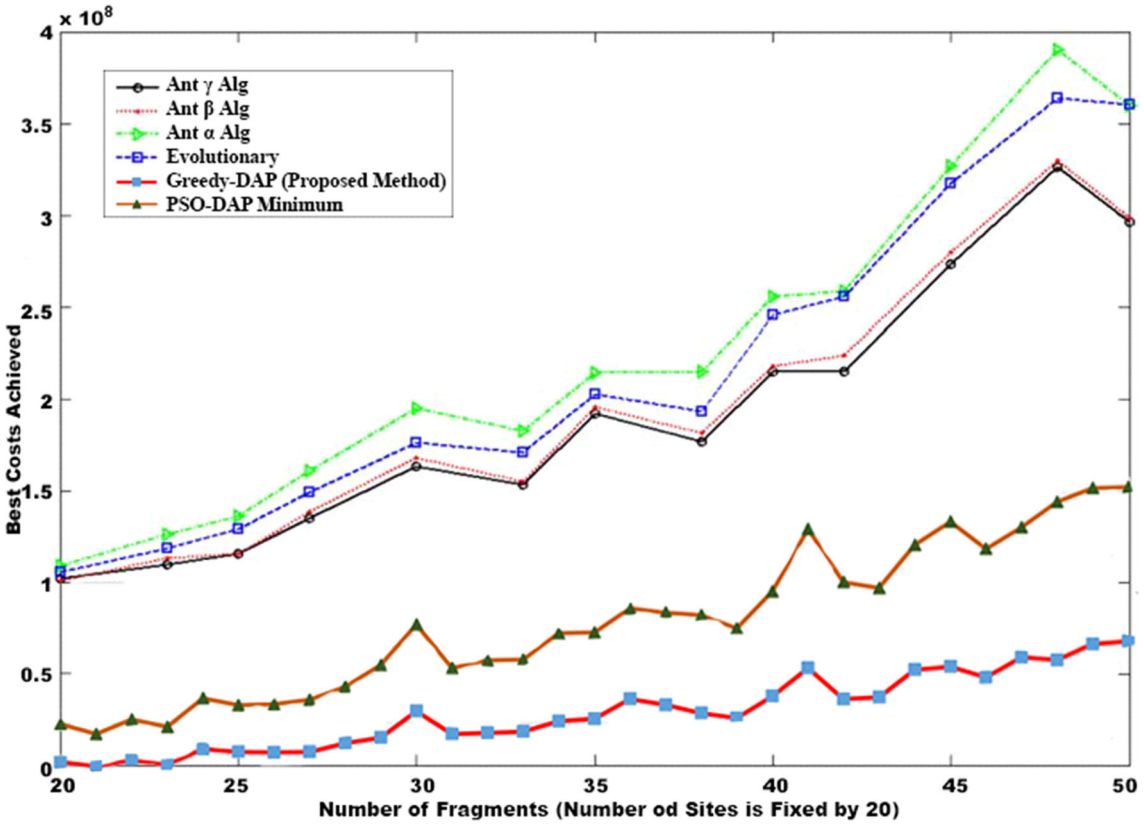

Fig. 6 Evaluating the Results achieved by the algorithms in a state 2 comparison for cost (Adl and Rankoohi 2009)

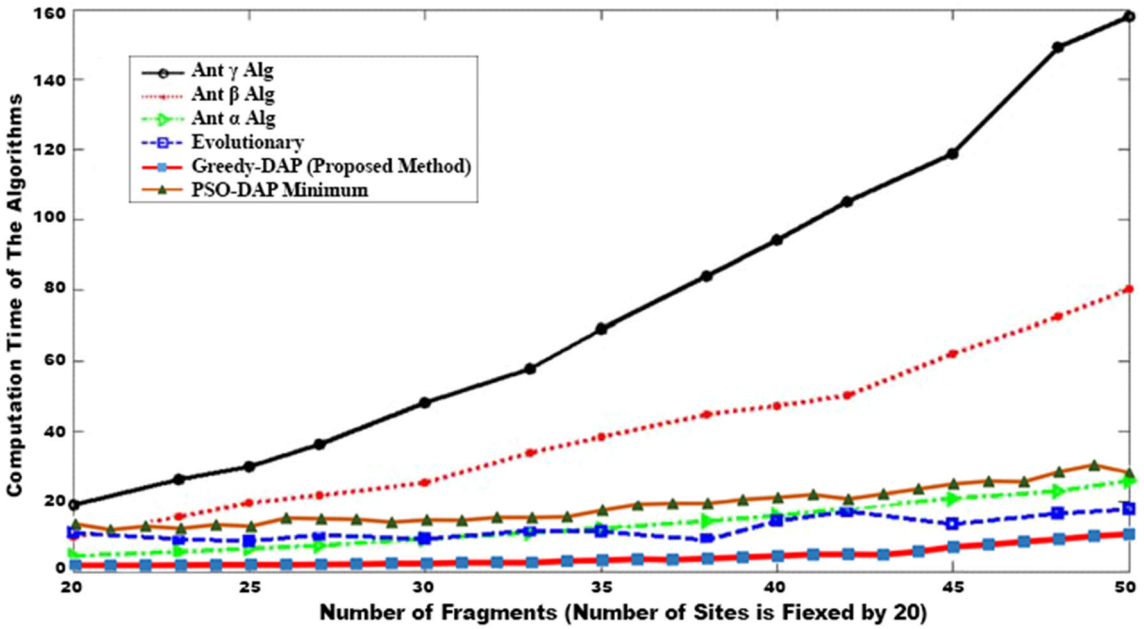

Fig. 7 Evaluating the Computation time of the algorithms in a state 2 comparison for time (Adl and Rankoohi 2009)

The statistical comparison of the proposed method and the other method is given in the last six rows of Table 16 for increasing the fragment numbers (m) and fixing the site number, showing that this algorithm has better results. To better show the results of the presented method, we have compared three data states. In the first state, we considered the number of sites varied and the number fragments fixed. For the second case, we considered the number of

Table 9 Generate cost and execution time for increasing fragment numbers (m) and site number is fixed by 20

m	Cost $\times 10^8$	Time (s)	M	Cost $\times 10^8$	Time (s)
20	0.268	0.961	36	0.690	5.343
21	0.227	1.215	37	0.686	6.132
22	0.278	1.189	38	0.674	7.018
23	0.254	1.384	39	0.655	7.775
24	0.348	1.566	40	0.758	8.390
25	0.364	1.590	41	0.951	9.807
26	0.342	1.894	42	0.803	10.488
27	0.351	1.864	43	0.823	12.391
28	0.419	2.362	44	0.955	14.276
29	0.523	2.450	45	1.083	14.880
30	0.586	2.721	46	0.978	18.289
31	0.485	3.290	47	1.025	19.571
32	0.515	3.820	48	1.135	20.931
33	0.507	4.167	49	1.114	24.177
34	0.649	4.160	50	1.178	25.570
35	0.621	5.017			

Table 10 Cost comparison of methods for increasing fragment numbers (m) and site number is fixed by 20

m	PSO-Ave	PSO-min	Cost Proposed Method	Standard deviation	m	PSO-Ave	PSO-min	Cost Proposed Method	Standard deviation
20	0.252	0.249	0.244	0.035	36	0.600	0.589	0.657	0.059
21	0.220	0.208	0.219	0.030	37	0.606	0.600	0.619	0.010
22	0.277	0.267	0.256	0.047	38	0.664	0.638	0.564	0.028
23	0.250	0.247	0.228	0.033	39	0.616	0.584	0.532	0.033
24	0.334	0.330	0.331	0.013	40	0.694	0.682	0.675	0.023
25	0.307	0.304	0.312	0.041	41	0.923	0.906	0.858	0.037
26	0.316	0.314	0.309	0.027	42	0.763	0.761	0.656	0.066
27	0.354	0.339	0.312	0.029	43	0.720	0.711	0.668	0.053
28	0.373	0.367	0.369	0.034	44	0.839	0.812	0.847	0.004
29	0.482	0.472	0.404	0.042	45	0.985	0.943	0.867	0.053
30	0.566	0.563	0.578	0.041	46	0.848	0.837	0.797	0.033
31	0.463	0.429	0.429	0.039	47	0.866	0.847	0.928	0.066
32	0.483	0.471	0.436	0.015	48	1.061	1.030	0.910	0.105
33	0.480	0.468	0.444	0.048	49	1.121	1.016	1.014	0.070
34	0.583	0.572	0.512	0.030	50	1.029	1.004	1.033	0.200
35	0.575	0.562	0.527	0.041					

fragments varied and the number of sites constant. For the third state, we considered the number of sites and the number of fragments to be the same in order to better show the correct performance of the presented method. In all three states, the results of the allocated cost and especially the time consumption of the presented method are much better than the previous algorithms.

Table 11 Execution time (s) comparison of methods for increasing fragment numbers (m) and site number is fixed by 20

m	PSO-DAP Minimum	Greedy-DAP (Proposed Method)	m	PSO-DAP Minimum	Greedy-DAP (Proposed Method)
20	3.111	0.218	36	3.472	2.028
21	3.422	0.203	37	3.686	1.888
22	3.113	0.265	38	4.146	2.168
23	2.814	0.281	39	3.564	2.558
24	3.091	0.374	40	3.741	2.902
25	3.507	0.421	41	3.300	3.370
26	3.197	0.437	42	3.399	3.416
27	3.856	0.499	43	4.352	3.292
28	3.984	0.608	44	3.774	4.243
29	4.593	0.796	45	3.259	5.538
30	4.290	0.764	46	4.037	6.193
31	3.988	0.998	47	3.540	7.004
32	4.678	1.076	48	3.862	7.738
33	3.444	1.030	49	3.372	8.658
34	2.995	1.466	50	3.187	9.142
35	3.532	1.685			

Table 12 For increasing fragment numbers (m) and site number is fixed by 20. Obtained results of methods comparisons with sign test

m	Greedy-DAP (Proposed method)	PSO-DAP minimum	Sign	m	Greedy-DAP (Proposed Method)	PSO-DAP minimum	Sign
20	24,433,882	24,872,829	+	36	65,701,506	58,860,638	+
21	21,919,467	20,829,479	−	37	61,854,394	60,038,953	−
22	25,623,071	26,714,534	+	38	56,381,250	63,792,446	−
23	22,791,175	24,670,355	−	39	53,194,842	58,383,603	−
24	33,091,691	33,007,551	+	40	67,450,888	68,192,047	−
25	31,215,154	30,397,092	−	41	85,806,612	90,561,561	+
26	30,898,685	31,405,235	−	42	65,616,347	76,080,351	−
27	31,228,117	33,875,190	−	43	66,842,906	71,074,916	−
28	36,943,665	36,689,174	−	44	84,666,662	81,243,023	−
29	40,435,858	47,213,491	−	45	86,716,618	94,333,685	+
30	57,775,641	56,279,930	+	46	79,691,135	83,669,594	−
31	42,873,844	42,876,940	−	47	92,754,465	84,678,495	−
32	43,620,635	47,103,581	−	48	91,001,545	103,032,353	−
33	44,429,550	46,819,696	−	49	101,370,565	101,586,213	+
34	51,215,186	57,234,406	−	50	103,267,955	100,401,042	+
35	52,734,960	56,175,376	−				

Statistical notations	Greedy-DAP vs PSO
	9
	15
	2.738
	− 2.191
H_0	Reject
H_1	Accept

Table 13 Generate cost and execution time for increasing DAP instance sizes

Size	Cost × 10^6	Time (s)	Size	Cost × 10^6	Time (s)
5	0.07	0.14	55	145.23	25.18
10	0.21	0.20	60	212.56	55.97
15	0.55	0.75	65	290.99	147.06
20	2.49	0.89	70	319.86	166.73
25	8.91	1.56	75	430.69	197.95
30	9.65	2.28	80	594.30	253.48
35	25.67	3.17	85	771.51	243.56
40	43.91	6.13	90	1047.78	323.75
45	73.52	10.83	95	1274.53	331.08
50	112.09	21.04	100	1487.04	337.76

Table 14 Cost comparison of methods for increasing DAP instance sizes (cost value is column × 10^6)

Size	ACO (Tosun 2014a)	RTS (Tosun 2014a)	GA1 (Tosun 2014a)	GA2 (Tosun 2014a)	GA3 (Tosun 2014a)	HG-MTS (Tosun 2014a)	SA (Tosun et al. 2013b)	PSO-DAP		DEVNS (Lotfi 2019)	Greedy-DAP (Proposed Method)
								Standard deviation	Average		
5	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.0007	0.03	0.03	0.003
10	0.31	0.31	0.32	0.31	0.31	0.31	0.31	0.0130	0.06	0.06	0.04
15	0.98	0.98	0.99	0.98	0.98	0.98	0.98	0.0532	0.52	0.52	0.14
20	2.61	2.61	2.63	2.64	2.64	2.61	2.61	0,0816	1.47	1.47	0.44
25	5.19	5.19	5.25	5.26	5.24	5.19	5.15	1.2826	3.35	3.35	0.60
30	10.27	10.27	10.39	10.42	10.41	10.27	10.27	1.0470	2.98	2.98	1.36
35	16.39	16.39	16.64	16.61	16.66	16.39	16.41	4.5046	8.51	8.51	1.96
40	25.91	25.9	26.28	26.33	26.21	25.92	26.02	8.9307	20.71	20.71	3.51
45	37.28	37.26	37.73	37.8	37.82	37.27	37.40	16.7220	25.56	25.56	5.54
50	53.93	53.89	54.76	54.63	54.69	53.88	54.08	25.6230	29.38	29.38	7.60
55	71.30	71.19	72.72	72.40	72.13	71.21	71.40	37.4836	46.19	46.19	34.91
60	90.35	90.16	91.76	91.49	91.56	90.20	90.50	59.9261	90.2	90.2	44.75
65	112.31	112.13	113.59	113.75	113.84	112.08	112.49	91.0824	113.72	113.72	68.88
70	146.41	146.19	148.48	148.8	148.18	146.15	146.73	109.8225	131.72	131.72	106.34
75	177.90	177.7	180.04	180.75	180.63	177.65	178.16	139.8139	168.34	168.34	163.89
80	219.40	219.26	223.10	222.80	222.96	219.18	219.81	160.2445	234.26	234.26	159.95
85	262.24	261.88	267.04	266.15	266.19	261.99	262.89	240.2334	260.33	260.33	196.66
90	316.11	315.86	320.88	320.93	320.58	315.86	316.81	302.8202	279.49	279.49	265.25
95	370.14	369.92	375.49	375.85	375.29	369.91	371.14	392.3170	355.04	355.04	353.27
100	428.40	428.28	436.19	436.15	434.45	427.98	429.10	459.9464	424.08	424.08	408.62

3.4 Discussion

The aims of the data allocation problem (DAP) are to achieve the minimum execution time and ensure the lowest transaction cost of queries. The solution for this NP-hard problem based on numerical methods is computationally expensive. Despite the success of such heuristic algorithms as GA and PSO in solving DAP, the initial control parameters tuning, the relatively high convergence speed, and hard adaptations to the problem are the most important disadvantages of these methods. This paper presents a simple well-formed greedy algorithm to optimize the total transmission cost of each site-fragment dependency and each inner-fragment dependency. To evaluate the effect of

Table 15 Execution time (s) comparison of methods for increasing DAP instance sizes

DAP Size	ACO (Tosun 2014a)	RTS (Tosun 2014a)	GA1 (Tosun 2014a)	GA2 (Tosun 2014a)	GA3 (Tosun 2014a)	HG-MTS (Tosun 2014a)	SA (Tosun et al. 2013b)	PSO-DAP(Mahi et al. 2018)	DEVNS (Lotfi 2019)	Greedy-DAP (Proposed Method)
5	9.26	0.83	76.27	56.11	88.11	1.44	130.29	0.74	65	0.004
10	14.52	2.73	87.80	60.37	94.91	2.45	143.84	1.35	60	0.01
15	13.74	5.66	90.76	66.22	104.13	2.65	214.30	2.17	58	0.02
20	17.91	8.89	123.79	84.13	167.22	4.17	243.30	3.65	52	0.08
25	25.86	14.52	131.98	81.96	125.30	5.21	351.23	4.25	42	0.34
30	31.17	20.89	132.46	104.64	137.02	7.38	461.89	6.45	24	5.05
35	43.31	29.06	150.06	111.87	151.02	10.73	393.73	6.81	21	12.89
40	56.59	37.05	166.80	128.75	173.21	15.60	420.65	8.85	30	32.06
45	80.92	48.67	191.93	159.10	202.10	20.80	437.74	8.42	10	67.78
50	105.33	62.74	471.98	207.56	359.57	26.80	511.40	9.60	4	134.00
55	126.00	76.07	268.31	201.43	261.71	27.22	516.86	13.74	3	335.31
60	166.55	91.79	315.31	208.37	290.46	39.56	828.14	16.09	46	659.49
65	204.35	109.20	421.93	284.08	336.01	48.92	1090.77	16.09	51	1015.80
70	320.62	131.54	536.15	344.20	358.03	63.13	1303.21	17.34	8	1868.15
75	309.51	155.31	609.77	379.07	380.81	73.41	976.97	17.01	11	2712.19
80	396.18	193.63	464.17	331.17	416.18	87.84	1234.48	16.29	174	3755.91
85	807.43	195.80	532.05	364.71	586.21	102.79	898.11	18.31	36	5516.20
90	621.55	215.58	563.15	400.37	531.13	123.19	1336.74	20.98	1	7898.36
95	725.93	250.72	629.55	974.24	569.92	143.16	1128.08	18.15	7	11,376.30
100	1203.99	278.63	1236.30	568.73	808.82	179.07	1389.19	20.97	9	15,350.07

the proposed method, more than 20 standard DAP problems were used. Experimental results showed that the proposed approach had better quality in terms of execution time and total cost.

4 Conclusion

In this work, we have solved DAP in non-replicated distributed database systems. The key subjective in DAP is to decrease the query execution time and the transaction cost. In this paper, as an approach based on the greedy algorithm, Greedy-DAP has been proposed to optimize query execution time and the transaction cost in DAP. To evaluate the effectiveness of the proposed algorithm, three states were considered. The first state referred to the set constant fragment size and variant site size. In addition, in the second state, the constant size of sites and variant sizes of fragments were considered. The third state contained a constant size of sites and fragments. The obtained results of the proposed algorithm based on these three states demonstrated that the presented method could have a good performance. Greedy-DAP was founded on various DAP samples in several states, obtaining effective results in comparison with other algorithms. Also, it was evaluated in terms of the query execution time and transaction cost. The obtained results demonstrated that Greedy-DAP displayed better performance in terms of query solution quality and running time. Due to the solution, space exponentially grows along with increasing the dimensionality of the problem; the performance of the approach is decreased. However, analysis of the results demonstrated that the presented approach generated results comparable with the state of art algorithms, especially in terms of execution time, due to the lower number of computations.

To solve the DAP problem that was previously solved with the PSO algorithm, it can be solved with a parallel algorithm, which can get much better results. Also, the

Table 16 Statistical comparison of the methods using sign test

Greedy-DAP (Proposed method)	DEVNS (Lotfi 2019)	Sign	PSO-DAP (Mahi et al. 2018)	Sign	ACO (Tosun 2014a)	Sign	RTS(Tosun 2014a)	Sign
0.003	0.03	–	0.02	–	0.04	–	0.04	–
0.04	0.06	–	0.05	–	0.31	–	0.31	–
0.14	0.52	–	0.41	–	0.98	–	0.98	–
0.44	1.47	–	0.77	–	2.61	–	2.61	–
0.6	3.35	–	3.74	–	5.19	–	5.19	–
1.36	2.98	–	3.19	–	10.27	–	10.27	–
1.96	8.51	–	9.04	–	16.39	–	16.39	–
3.51	20.71	–	19.24	–	25.91	–	25.9	–
5.54	25.56	–	27.04	–	37.28	–	37.26	–
7.6	29.38	–	34.43	–	53.93	–	53.89	–
34.91	46.19	–	51.38	–	71.3	–	71.19	–
44.75	90.2	–	97.78	–	90.35	–	90.16	–
68.88	113.72	–	125.01	–	112.31	–	112.13	–
106.34	131.72	–	138.69	–	146.41	–	146.19	–
163.89	168.34	–	171.47	–	177.9	–	177.7	–
159.95	234.26	–	260.86	–	219.4	–	219.26	–
196.66	260.33	–	260.63	–	262.24	–	261.88	–
265.25	279.49	–	287.09	–	316.11	–	315.86	–
353.27	355.04	–	365.06	–	370.14	–	369.92	–
408.62	424.08	–	481.58	–	428.4	–	428.28	–

Statistical Notations	Greedy-DAPvs. DEVNS	Greedy-DAPvs. PSO	Greedy-DAPvs. ACO	Greedy–DAPvsRTS
S	0	0	0	0
X_s	10	10	10	10
σ_s	2.236	2.236	2.236	2.236
Z	− 4.472	− 4.472	− 4.472	− 4.472
$Z_{\frac{0.05}{2}}$	± 1.96	± 1.96	± 1.96	± 1.96
H0	Reject	Reject	Reject	Reject
H1	Accept	Accept	Accept	Accept

Greedy–DAP (Proposed Method)	GA1(Tosun 2014a)	Sign	GA2(Tosun 2014a)	Sign	GA3(Tosun 2014a)	Sign	HG–MTS (Tosun 2014a) (Tosun 2014a)	Sign	SA(Tosun et al. 2013b)	Sign
0.003	0.04	–	0.04	–	0.04	–	0.04	–	0.04	–
0.04	0.32	–	0.31	–	0.31	–	0.31	–	0.31	–
0.14	0.99	–	0.98	–	0.98	–	0.98	–	0.98	–
0.44	2.63	–	2.64	–	2.64	–	2.61	–	2.61	–
0.6	5.25	–	5.26	–	5.24	–	5.19	–	5.15	–
1.36	10.39	–	10.42	–	10.41	–	10.27	–	10.27	–
1.96	16.64	–	16.61	–	16.66	–	16.39	–	16.41	–
3.51	26.28	–	26.33	–	26.21	–	25.92	–	26.02	–
5.54	37.73	–	37.8	–	37.82	–	37.27	–	37.4	–
7.6	54.76	–	54.63	–	54.69	–	53.88	–	54.08	–
34.91	72.72	–	72.4	–	72.13	–	71.21	–	71.4	–
44.75	91.76	–	91.49	–	91.56	–	90.2	–	90.5	–
68.88	113.59	–	113.75	–	113.84	–	112.08	–	112.49	–
106.34	148.48	–	148.8	–	148.18	–	146.15	–	146.73	–
163.89	180.04	–	180.75	–	180.63	–	177.65	–	178.16	–

Table 16 (continued)

Greedy–DAP (Proposed Method)	GA1(Tosun 2014a)	Sign	GA2(Tosun 2014a)	Sign	GA3(Tosun 2014a)	Sign	HG–MTS (Tosun 2014a) (Tosun 2014a)	Sign	SA(Tosun et al. 2013b)	Sign
159.95	223.1	–	222.8	–	222.96	–	219.18	–	219.81	–
196.66	267.04	–	266.15	–	266.19	–	261.99	–	262.89	–
265.25	320.88	–	320.93	–	320.58	–	315.86	–	316.81	–
353.27	375.49	–	375.85	–	375.29	–	369.91	–	371.14	–
408.62	436.19	–	436.15	–	434.45	–	427.98	–	429.1	–

Statistical Notations	Greedy-DAPvs. GA1	Greedy-DAPvs. GA2	Greedy-DAPvs. GA3	Greedy-DAPvs. HG-MTS	Greedy-DAPvs. SA
S	0	0	0	0	0
X_s	10	10	10	10	10
σ_s	2.236	2.236	2.236	2.236	2.236
Z	− 4.472	− 4.472	− 4.472	− 4.472	− 4.472
$Z_{\frac{0.05}{2}}$	± 1.96	± 1.96	± 1.96	± 1.96	± 1.96
H0	Reject	Reject	Reject	Reject	Reject
H1	Accept	Accept	Accept	Accept	Accept

gray wolf optimization (GWO) algorithm, the firefly optimization algorithm, the grasshopper optimization algorithm (GOA), the cuckoo optimization algorithm, and the frog leap optimization algorithm can be used in the future.

Acknowledgements This study has been supported by Payame Noor University.

Author contributions MM was involved in the conceptualization, methodology, coding, and writing, OKB contributed to the methodology, investigation, writing. HK assisted in the supervision, methodology, reviewing and editing.

Funding The authors have not disclosed any funding.

Data availability The data that support the findings of this study are available from the corresponding author on request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Informed consent Informed consent was obtained from all authors included in the study. This manuscript does not contain any studies with human participants or animals performed by any of the authors.

References

Adl RK, Rankoohi SMTR (2009) A new ant colony optimization based algorithm for data allocation problem in distributed databases. Knowl Inf Syst 20(3):349–373

Al-Sanhani AH et al. (2017) *A comparative analysis of data fragmentation in distributed database*. In: Information technology (ICIT), 2017 8th International conference on. 2017. IEEE

Amer AA, Sewisy AA, Elgendy TM (2017) An optimized approach for simultaneous horizontal data fragmentation and allocation in distributed database systems (DDBSs). Heliyon 3(12):e00487

Amer AA, Abdalla HI (2012) A heuristic approach to re-allocate data fragments in DDBSs. In: Information technology and e-services (ICITeS), 2012 international conference on. 2012. IEEE

Astrachan OL et al. (2002) Active learning in small to large courses. In: 32nd annual frontiers in education. IEEE

Bai Q (2010) Analysis of particle swarm optimization algorithm. Comput Infor Sci 3(1):180–184

Cao X (2022) Goals and solutions of data allocation in data center. In: 2022 IEEE 12th annual computing and communication workshop and conference (CCWC), IEEE

Chen W et al (2018) A cost minimization data allocation algorithm for dynamic datacenter resizing. J Parallel Distrib Comp 118:280–295

Deng W et al (2012) A novel parallel hybrid intelligence optimization algorithm for a function approximation problem. Comput Math Appl 63(1):325–336

Du J et al (2017) Optimization of data allocation on CMP embedded system with data migration. Int J Parallel Prog 45(4):965–981

Gu X, Lin WJ, Veeravalli B (2006) Practically realizable efficient data allocation and replication strategies for distributed databases with buffer constraints. IEEE Trans Parallel Distrib Syst 17(9):1001–1013

Kadam S, Kim DI (2022) Knowledge-aware semantic communication system design and data allocation. arXiv preprint arXiv:2301.03468

Li Z, Chen X, Han Y (2022) Optimal data allocation for graph processing in processing-in-memory systems. In: 2022 27th Asia and south pacific design automation conference (ASP-DAC), IEEE

Lotfi N (2019) Data allocation in distributed database systems: a novel hybrid method based on differential evolution and variable neighborhood search. SN Appl Sci 1(12):1724

Lurie D, Abramson LR, Vail JA (2011) *Applying statistics*. 2011: Citeseer

Mahi M, Baykan OK, Kodaz H (2015) A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem. Appl Soft Comput 30:484–490

Mahi M, Baykan OK, Kodaz H (2018) A new approach based on particle swarm optimization algorithm for solving data allocation problem. Appl Soft Comput 62:571–578

Mamaghani AS et al. (2010) A novel evolutionary algorithm for solving static data allocation problem in distributed database systems. In: Network applications protocols and services (NETAPPS), 2010 Second International Conference on. 2010. IEEE

Mann PS (2013) Introductory Statistics, 8th edn. Wiley, Hoboken

Mashwani WK, Salhi A (2012) A decomposition-based hybrid multiobjective evolutionary algorithm with dynamic resource allocation. Appl Soft Comput 12(9):2765–2780

Mathew AB (2018) Data allocation optimization for query processing in graph databases using Lucene. Comput Electr Eng 70:1019–1033

Mayne SR, Satav S (2017) Survey on cloud infrastructure resource allocation for big data applications. Int J Eng Sci 7(1):4008

Nashat D, Amer AA (2018) A comprehensive taxonomy of fragmentation and allocation techniques in distributed database design. ACM Comp Surv (CSUR) 51(1):12

Peng C et al. (2022) Optimal data allocation in the environment of edge and cloud servers. In: 2022 IEEE International conference on networking, sensing and control (ICNSC)

Sen G et al (2016) Mathematical models and empirical analysis of a simulated annealing approach for two variants of the static data segment allocation problem. Networks 68(1):4–22

Thalij SH (2022) Data allocation in distributed database based on CSO. Tikrit J Pure Sci 27(2):43–51

Tosun U (2014a) Distributed database design using evolutionary algorithms. J Commun Netw 16(4):430–435

Tosun U (2014b) A new recombination operator for the genetic algorithm solution of the quadratic assignment problem. Procedia Comp Sci 32:29–36

Tosun U, Dokeroglu T, Cosar A (2013a) A robust island parallel genetic algorithm for the quadratic assignment problem. Int J Prod Res 51(14):4117–4133

Tosun U, Dokeroglu T, Cosar A (2013) Heuristic algorithms for fragment allocation in a distributed database system. Computer and Information Sciences, vol III. Springer, Berlin, pp 401–408

Ulus T, Uysal M (2003) Heuristic approach to dynamic data allocation in distributed database systems. Pakistan J Inf Technol 2(3):231–239