Applied Soft Computing 62 (2018) 571–578

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Full Length Article

A new approach based on particle swarm optimization algorithm for solving data allocation problem

Mostafa Mahi[a], Omer Kaan Baykan[b], Halife Kodaz[b,*]

[a] Department of Computer Engineering and Information Technology, Payame Noor University, PO Box 19395-3697 Tehran, Iran
[b] Department of Computer Engineering, Faculty of Engineering, Selcuk University, Konya, Turkey

ARTICLE INFO

Article history:
Received 1 December 2016
Received in revised form 18 October 2017
Accepted 12 November 2017
Available online 21 November 2017

Keywords:
Data allocation problem
Particle swarm optimization
Distributed databases system
Site-fragment dependency

ABSTRACT

The effectiveness distributed database systems highly depends on the state of site that its task is to allocate fragments. This allocation purpose is performed for obtaining the minimum execute time and transaction cost of queries. There are some NP-hard problems that Data Allocation Problem (DAP) is one of them and solving this problem by means of enumeration method can be computationally expensive. Recently heuristic algorithms have been used to achieve desirable solutions. Due to fewer control parameters, robustness, speed convergence characteristics and easy adaptation to the problem, this paper propose a novel method based on Particle Swarm Optimization (PSO) algorithm which is suitable to minimize the total transmission cost for both the each site – fragment dependency and the each inter – fragment dependency. The core of the study is to solve DAP by utilizing and adaptation PSO algorithm, PSO-DAP for short. Allocation of fragments to the site has been done with PSO algorithm and its performance has been evaluated on 20 different test problems and compared with the state-of-art algorithms. Experimental results and comparisons demonstrate that proposed method generates better quality solutions in terms of execution time and total cost than compared state-of-art algorithms.

1. Introduction

One of the most crucial applications that recently have attracted attentions is distributed database such as Data Allocation Problem (DAP). The purpose of the DAP is to determine a placement of fragments at the best different sites in order to minimize the total transaction cost when a query is taken from one site to another. DAP refers to a standard test problem used in performance analysis of optimization algorithms with certain constraints [1]. Nowadays it is extremely important to efficiently allocate data to sites. In the real world, the data used by search engines or mail servers are very large and disorganized. The locations of those who request data can change. In this case, it is necessary to efficiently organize the data. For example some issues such as parallel query execution, network load and servers load balancing are design factors that need to be handled. In the real world without considering of mentioned parameters, DAP is an NP-hard problem. Two types of static and dynamic algorithms are presented to solve DAP. Static algorithm refers to data allocation based on static transaction execution pattern in the target environment whereas these patterns can be changed in dynamic algorithms [2–5].

There are some similarities between a Quadratic Assignment Problem (QAP) with a DAP, such that, it keeps track of the resource locality. A QAP is developed by Koopmans and Beckman [6] and is one of the fundamental combinatorial optimization problems. QAP refers to the dependency of sites and fragments, the sites are pointed as locations and the fragments as facilities.

There is several solutions to solve DAP and QAP problems. Some solution methodologies are detailed in related works (Section 2). Among the previous solution methodologies developed for solving DAP and QAP, Particle Swarm Optimization (PSO) algorithm has been used due to its speed convergence characteristics, solution quality, robustness and easy adaptation to solve the problem in this study. There is no overlapping and mutation calculation over PSO algorithm. In the PSO-DAP method, search operation is performed by the speed of particle and during several generations, only the most optimist particles can transmit information to others. In this case researching speed will be so high [7,8].

In this study, we focus on solving DAPs by utilization and adaptation of PSO. According to the literature review, PSO has not been used to solve DAPs. Due to fact that PSO has fewer control parameters, speed convergence characteristics and lower consuming time,

* Corresponding author.
E-mail address: hkodaz@selcuk.edu.tr (H. Kodaz).

https://doi.org/10.1016/j.asoc.2017.11.019
1568-4946/© 2017 Elsevier B.V. All rights reserved.

robustness against to solution space of the optimization problems, it is frequently used to solve optimization problems with different characteristics by the practitioners and researchers [7,8]. In this study, DAP which is NP-hard problem is solved by PSO and the performance of the proposed algorithm is compared with the performance of genetic algorithm [1], tabu search [1], ant colony algorithm [1] and simulated annealing [9] on solving 20 problems with different dimensionalities. The execution time is important indicator and factor for the problem like total cost. The proposed algorithm produced the acceptable and comparable results in a reasonable time when we compare the execution time of PSO-DAP with other algorithms, the best results belong to proposed PSO-DAP. Additionally, a new dataset is prepared for comparison of future studies.

In Section 1, the study is introduced and the rest of the paper is organized as follow: The related works and methodologies for solving DAP and QAP are given in Section 2. Section 3 refers to material and methods that include background information for PSO algorithm, and presents the proposed method (PSO-DAP). Experiments and comparison are conducted in Section 4 and finally the study is concluded in Section 5.

2. Related works

In this section, some studies proposed for solving DAP and QAP are reviewed and summarized. As mentioned earlier, DAP and QAP are known as NP-hard problem and various meta heuristic methods such as Genetic Algorithm (GA) [10,11] and Ant Colony Optimization (ACO) algorithm [1,2] for solving these problems. Other studies are as follow:

Sen et al. proposed a Simulated Annealing (SA) method to solve data allocation problem [12]. They have analyzed the SA approach with benchmarks that are obtained from CPLEX benchmarks.

Wang et al. have modeled the Radio-Frequency Identification (RFID) tag oriented data allocation problem as a nonlinear knapsack problem [13]. They have utilized the artificial immune network (DA − aiNet) to address it. Some numerical assessments have been done to illustrate the effect of memory capacity and correlation matrix. Also, they have compared it with other present methods.

A set of Teaching Learning Based Optimization (TLBO) methods based on hybrid algorithms to solve the challenging combinatorial optimization QAP is suggested by Dokeroglu. Recombination operators and later a Robust Tabu-search engine that processing them, were used to train individuals. Sequential and parallel TLBO-based hybrid algorithms efficiency was compared with those of the mentioned heuristics in terms of the best solution and computational effort. The performance of proposed algorithms demonstrate that they are competitive with the best algorithms for the solution of the QAP with which many real life problems could be modeled [14].

A novel algorithm for replicated fragment allocation during distributed database design for static environment using Biogeography-Based Optimization (BBO) is presented by Singh et al. Mentioned algorithm targeted to design a replicated fragments method to minimize the sum of data transmission cost and storage cost of fragments. The results of biogeography – based optimization algorithm for data allocation were compared with GA to consider the effectiveness of the technique [15].

A new recombination operator based on Order-1 crossover algorithm was suggested by Tosun which runs the quick sort partitioning algorithm to produce various chromosomes for partitions. Offspring's with the other chromosomes are generated with the minimum cost partition [16]. The GA, the fast ACO and the robust Tabu-search for solving the DAP are used by Tosun. The presented method efficiency for sample size smaller than 50 with respect to the optimal results proposed in QAPLIB is outstanding.

Li and Wong have used time series modeling approach to predict the shot-term load [17]. In this approach the node number adjustment and fragment reallocation is considered to omit node overloading's and fragment reallocation originated from fragment mitigations.

In 2013, Tosun et al. have proposed a collection of SA, GA and Fast ACO to solve DAP [9]. Some GA crossover operator for QAP is explained by Tosun et al. and their effectiveness is assessed using well-known samples of QAPLIB library. The parameters of GA (such as the number of processing units, size of initial population and generations) are set similar to the best solutions for large QAP samples described in the library. The solution batches on each processing clusters are separated to design an efficient parallel GA using an island model. At the first level of optimization process, each processors solutions trade ten percent rate. By using the island Parallel GA, there are improvement in both run times and goodness of the solutions for large QAP samples [10].

Abdalla, by bringing a change to data access pattern, proposed a new data re-allocation model for replicated and non-replicated constrained Distributed Database Systems (DDBSs). Mentioned method assumption is based on a distribution of fragments over network sites that were done according to a properly forecasted set of query frequency values which could be employed over sites that takes sites' restrictions into account in the re-allocation phase. The proposed approach is an efficient plan to re-allocate data fragments across sites based on communication and update cost values for each fragment individually. The re-allocation process was founded on selecting the maximal update cost value for each fragment and making the re-allocation accordingly. Results demonstrate that mentioned algorithm act as an efficient algorithm in solving fragments re-allocation problem in a dynamic distributed relational databases environment [18].

Another data re-allocation model for DDBSs by altering procedure of data access over sites is proposed by Amer and Abdalla, which assumed that the scattering of fragments over network sites was initially achieved according to an appropriately predicted collection of query frequency values that could be employed over sites. In their assumption a plan to re-allocate data fragments based on communication costs between sites and update cost values for each fragment is very important. The re-allocation process was based on selecting the maximum update cost value for each fragment and deciding on the re-allocation accordingly. It is confirmed that the proposed technique will effectively contribute in solving dynamic fragments re-allocation problem in the context of a distributed relational database systems [19].

In 2009, Adl and Rankoohi have proposed ACO-DAP model based on the ACO and a local search [2]. Another heuristic GA for solving Resource Allocation Problems (RAPs) was proposed by Lee et al. in 2003 [20]. Its objective was to overcome RAPs which are appeared in practice. Their studies were based on several genetic algorithms and simulation results confirmed that the proposed algorithm had a good performance. A novel dynamic DDBS, called the threshold algorithm was proposed by Ulus and Uysal in 2003 [21]. This algorithm performs data reallocation via changing data access procedure. Simulation was a means to analyze a fragment. Where data access pattern changed dynamically, threshold algorithm was suitable for the DDS.

In 2002, Ahmad and Karlapalem have modeled the dependency between the fragments achieved by a query using a dependency graph. The graph has been utilized to formulate and tackle DAPs for distributed database systems based on query-site and move-small query execution strategies [22]. Evolutionary algorithms for data allocation in distributed database systems were designed and evaluated in their algorithm.

As a result of the literature review, new methods are needed to increase the cost and time efficiency of the methods proposed for

DAP. According to the literature review, PSO has not been used to solve DAPs. In this study, we focus on solving DAPs by utilization and adaptation of PSO. In the experiments, the execution times and the quality of the fragment allocation alternatives are investigated by using PSO. The proposed algorithm produced the acceptable and comparable results in a reasonable time when we compare the execution time of PSO-DAP with other algorithms.

3. Materials and methods

This section contains background information for PSO algorithm, solution information for DAP and the PSO-DAP. Description of notations is given in Table 1.

3.1. Particle swarm optimization algorithm

Kennedy and Eberhart in 1995 developed an optimization algorithm called PSO. PSO comprised of individuals that are called a particle and pinpoints to a solution in search space. The particle size varies according to the problem size. First, particles are determined randomly and each of them has an initial velocity. Each particle evaluation depends on fitness function. velocities of particle's are updated based on global best solution as in Eq. (1) and their locations are updated based on global best solution as in Eq. (2) [23–25].

$$v_i^{t+1} = v_i^t + c_1 r_1^t \left(pbest_i^t - X_i^t\right) + c_2 r_2^t \left(gbest - X_i^t\right) \quad (1)$$

$$X_i^{t+1} = X_i^t + v_i^{t+1} \quad (2)$$

Where X_i^t is location of the i^{th} particle in iteration t, X_i^{t+1} is location of i^{th} particle in iteration $t+1$, v_i^t is velocity vector of i^{th} particle in iteration t, v_i^{t+1} is velocity vector of i^{th} particle in iteration $t+1$. c_1 and c_2 are fixed numbers used for calculate new velocity based on particle's solution particle best (pbest) and global best solution global best (gbest). c_1, c_2 can be at interval $[0, 4]$. r_1^t and r_2^t are random numbers at interval $[0, 1]$ in iteration t. Also, inertia weight w is used in improved PSO versions. It is used to determine the impact of old velocity vector on new ones. It was added to Eq. (1) [26]. PSO algorithm works until it reaches a certain number of iterations.

3.2. Data allocation problem

The purpose of the DAP is to determine a placement of fragments at the best different sites so as to minimize the total transaction cost when a query is taken from one site to another [2,27]. Firstly, a dataset is required to solve the DAP. The dataset contains n sites,

Table 1
Description of notations [2].

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 i^{th} 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 j^{th} fragment.
$fragSize_j$	The size of fragment.
L	The number of considered transactions
t_k	The k^{th} 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 site s_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 transaction t_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 schemeΨ.
$COST\,(\Psi)$	The cost of data transmission in an allocation schemeΨ.
$COST1\,(\Psi)$	The cost of data transmission in an allocation schemeΨ 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 si 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
V	The velocity of Particle
X	The position of Particle
$TotalCap_i$	The current capacity of site S_i
$IterationNumber$	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.

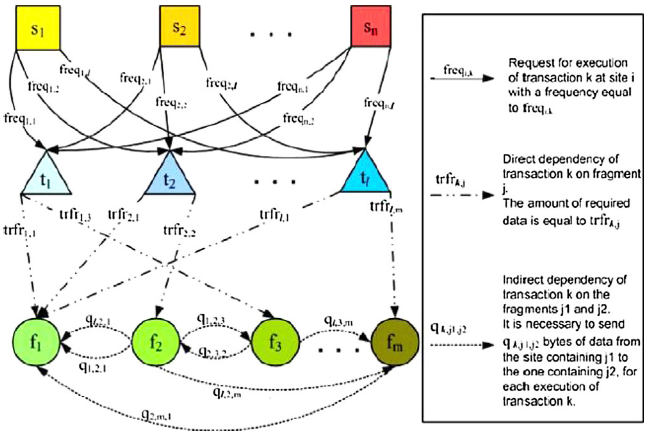
s1
s2
sn
freq1,l
freq1,2
freq1,1
freq2,1
freq2,2
freq2,l
freqn,1
freqn,2
freqn,l
t1
t2
tl
trfr1,1
trfr1,3
trfr2,1
trfr2,2
trfrl,1
trfrl,m
f1
f2
f3
fm
ql,2,1
q1,2,1
q1,2,3
q2,3,2
ql,3,m
ql,2,m
q2,m,1
freqi,k
Request for execution of transaction k at site i with a frequency equal to freqi,k
trfrk,j
Direct dependency of transaction k on fragment j.
The amount of required data is equal to trfrk,j
qk,j1,j2
Indirect dependency of transaction k on the fragments j1 and j2.
It is necessary to send qk,j1,j2 bytes of data from the site containing j1 to the one containing j2, for each execution of transaction k.

Fig. 1. The dependences among sites, transactions and fragments [2].

m fragments and *l* transactions. The dataset containing sites, fragments and transactions is constructed according to the following formulas.

Sites, transactions and fragments dependencies are shown in Fig. 1. Fig. 1 are taken directly from [2]. For instance, transaction k is required in order to get query from S_1 to S_2 to achieve fragment j.

Site Fragment Frequency (FREQ) matrix contains frequency values among sites transactions and is used in order to access transactions to website. Consequently Transactions to Fragmentation (TRFR) matrix is used in order to access transactions to fragmentation.

TRFR matrix includes some parameters such as relationship of transactions amount data which are required for fragments dependency. Q matrix refers to data between two fragments of a transaction. Each fragments size is at interval $\left[\frac{c}{10}, \frac{20*c}{10}\right]$ which is chosen randomly. In this interval $\left[\frac{c}{10}, \frac{20*c}{10}\right]$ c is a value at interval [10, 1000][2].

Among fragment sizes which are generated randomly, the highest one is chosen and assigned to maximum variable. An array which its length is m and elements (p_i) produced randomly between $\left[1, \left(2*\left(\frac{m}{n}\right)\right)-1\right]$ whereas during generation period the overall obtained amount should be equal to m($\sum_{i=1}^{n} p_i = m$. Remaining fragments (rf_i) are calculated using this formula: $rf_i = m - \sum_{q=1}^{i} p_q$

We use Eq. (3) to determine the site capacities [2].

$$siteCap_i = p_i * max_{1 \le j \le m}\left(fragSize_j\right) \tag{3}$$

i_1 and i_2 are transmission cost between two sites that are generated randomly at interval $UC_{i1,i2}$ on $[UCN, n*UCN]$.

Matrix FREQ's request frequency of each transaction in each site, has been denoted and n rows for sites and l columns for transactions are designated. The operation is executed in a way that first, 0, 1 are created randomly and consequently, for each of these 1's, random numbers between 1 and 1000 are produced.

It is worth to note that, in this case transaction access to the site is possible. In order to obligate a transaction access to the site, all of the FREQ matrix's elements must be multiplied to Request Probability of Transaction (RPT) (transaction being requested at a site factor). RPT is chosen randomly at 0 < RPT < = 1.

The TRFR matrix task is to denote the indirect transaction-fragment dependency and take into account the execution frequencies of the transactions. In this matrix rows are shown transactions and the columns are related to fragments. In the proposed approach 0 or 1 are generated randomly. Subsequently random numbers between 0 and fragment size for the 1 s are produced, and then we multiply the matrix with Access Probability to Fragment (APF). APF is chosen randomly at 0 < APF < = 1.

Q matrix denoting the indirect transaction-fragment dependency taking the execution frequencies of the transactions into account. In three-dimensional Q matrix rows and columns are fragments, and height is transaction. In order to determine matrix's element's value, we generated randomly numbers as 0 or 1. Then we again generated randomly numbers between 0 and fragment size for 1's, and we multiply the matrix to probability of a transaction necessitates data transmission between two sites (APFS).

In order to assign fragments to the sites randomly, matrix X(n, m)with random elements as 0 or 1's is created whereas each column is generated just for one iteration. By using matrix X, vector W with length m is created. Its elements refer to site's number that identifies which fragment will be placed at which site. The condition of this fact is that fragments that are allocated to sites shouldn't be overflowed site capacity.

$$X_{i,j} = \begin{bmatrix} 1 & \cdots & m \\ \vdots & \ddots & \vdots \\ n & \cdots & n*m \end{bmatrix}$$

The condition can be expressed as Eq. (4).

$$\sum_{j=1}^{m} fragSize_j * x_{ij} \le siteCap_i i = 1, 2, \ldots, n \tag{4}$$

COST1 represents the amount of site-fragment dependencies. Now for calculate COST1, we generate matrix STFR (The dependency of fragments on the site matrix).

$STFR_{n*m} = FREQ_{n*l} * TRFR_{l*m}$ or $stfr_{ij} = \sum_{k=1}^{l} freq_{ik} * trfr_{kj}$. From STFR matrix, PartialCost1 matrix is calculated by using Unit Transmission Cost (UC). PartialCost1 matrix has obtained by multiplying the UC$n*n$ on the STFR matrix as Eq. (5).

$$PARTIALCOST1_{n*m} = UC_{n*n} * STFR_{n*m} \tag{5}$$

In other words as Eq. (6).

$$partialcost1_{ij} = \sum_{q=1}^{n} uc_{iq} * stfr_{qj} \tag{6}$$

Finally, COST1 is calculated by using the Partial Cost on allocation vector (Eq. (7)).

$$COST1\left(\psi\right) = \sum_{j=1}^{m} partialcost1_{\psi j^{j}} \tag{7}$$

Matrix Q is used to calculate another cost term COST. In order to calculate the qfr_{kj1j2} matrix, we multiply sum of the kth column with the kth element of freq matrix to q_{kj1j2} matrix (Eq. (8)).

$$qfr_{kj1j2} = q_{kj1j2} * \sum_{r=1}^{n} freq_{kr} \tag{8}$$

Now we obtain the FRDEPmatrix by accumulate the cost of the transition between fragments (Eq. (9)).

$$frdep_{j1j2} = \sum_{k=1}^{l} qfr_{kj1j2} \tag{9}$$

COST2 is obtained by multiply FRDEP matrix and vector UC as Eq. (10).

$$\mathrm{COST2}\left(\psi\right)=\sum_{j1=1}^{m}\sum_{j2=1}^{m}\mathrm{frdep}_{j1j2}*\mathrm{uc}_{\psi_{j1}\psi_{j2}} \tag{10}$$

Finally, the total COSTis obtained with sum of COST2 and COST1 as Eq. (11). COST value is calculated based on produced vector. This paper is targeted to find the vector which is giving the lowest cost. In the next section, this goal will be achieved by our novel approach.

$$COST\left(\psi\right)=COST1\left(\psi\right)+COST2\left(\psi\right) \tag{11}$$

3.3. Proposed method (PSO-DAP)

In this study, we focus on solving DAPs by utilization and adaptation of PSO. According to the literature review, PSO has not been used to solve DAPs. Due to fact that PSO has fewer control parameters, speed convergence characteristics and lower consuming time, robustness against to solution space of the optimization problems, it is frequently used to solve optimization problems with different characteristics by the practitioners and researchers [7,8].

In the proposed method in order to solve DAP, we use PSO algorithm. We determined in each site which fragments are placed. Due to this fact that in this transaction the total cost must be low, in order to reduce the total cost of transaction, fragments are put at the best sites. In this paper to determine how fragments placed in the sites, PSO algorithm is used. Fragments replacements in the DAP is calculated by means of cost formula. In the PSO-DAP, the size of particle vector is 1*xm* and its structure is as follow:

		1	2	3	4	5	.	.	.	m	$m \to$ *fragment number*
P_k	= [	2	3	3	1	1	.	.	.	n]	$n \to$ *site number of the fragment*

P_k is kth particle, $P_k[j]=i$indicates that the jth fragment is located in the ith site ($i=1,2,..,n. j=1,2,..\ ,m$). For example, because of $P_k[1]=2$, fragment 1 is located in the site 2; because of $P_k[2]=3$, fragment 2 is located in the site 3 and etc. Particle shows that which fragment will be placed in which site.

To solve DAP; fragments placement on the best site by minimizing total transaction cost is our aim. In this paper, the process of fragments' allocation to sites has been evaluated by using PSO algorithm along with considering the total cost. For PSO algorithm, fitness is calculated by using cost function that is mentioned in Section 3.2. Pbest and gbest are determined by particle's cost values.

In the PSO-DAP, each site has a capacity. A counter is defined and used for each site to check the capacity of the site. Consequences of particles production, site capacities are checked. If there is an overcapacity in any site, the start locations of the particles are reproduced. Description of notations is given in [2].

If capacity of site is not exceed, the new positions of the particles are calculated using velocity. The initial velocity of each particle is produced randomly at interval[1, n]. If particle's location is beyond of 1 and n interval, these locations are randomly re-generated. As updating the particle's location, the obtained continuous values are rounded to the nearest integer. At the end of each iteration, pbest and gbest values are updated in the PSO algorithm. To stop the PSO-DAP, the maximum number of iteration is used. Pseudo-code, which belongs to the PSO-DAP, is shown in Fig. 2 and the scripting chart of the PSO-DAP is shown in Fig. 3.

4. Experimental results

To compare PSO-DAP with the alternative algorithms, first, the original data set is generated by using cost formulation in section 3.2. We assumed that the number of fragments and sites are equal due to this fact that our compared study is taken on an equal

```
- Initialization (number of particle, number of fragment and site, iteration number, length of the particle vector, site capacities, k = 1 )
Generate data set
Determination of the starting positions of the particles
While (k <= iteration number)
    - Calculation of cost
                    $COST(\psi) = COST1(\psi) + COST2(\psi)$
     - obtaining the value of pbest and gbest
     - calculating particles velocity.
          $V_i(k+1) = w*V_i(k) + c_1*rand_1*(pbest_i - x_i(k) + c_2*rand_2*(gbest - x_i(k))$
     -updating the location of particles
          $X_i(k+1) = X_i(k) + V_i(k+1)$
     If site capacities are overflow then generate new particle
          $k = k+1$
End while
```

Fig. 2. Pseudo-code of the PSO-DAP.

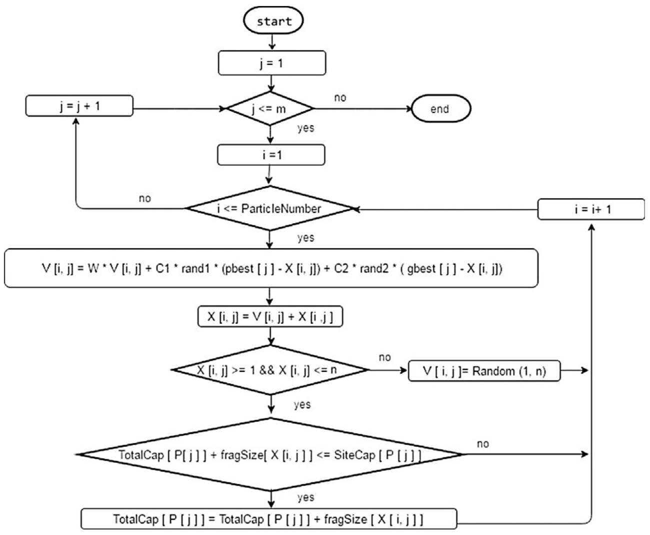
start
j = 1
j <= m
no
end
yes
j = j + 1
i =1
no
i <= ParticleNumber
i = i+ 1
yes
V [i, j] = W * V [i, j] + C1 * rand1 * (pbest [j] - X [i, j]) + C2 * rand2 * (gbest [j] - X [i, j])
X [i, j] = V [i, j] + X [i ,j]
X [i, j] >= 1 && X [i, j] <= n
no
V [i, j]= Random (1, n)
yes
TotalCap [P[j]] + fragSize[X [i, j]] <= SiteCap [P [j]]
no
yes
TotalCap [P [j]] = TotalCap [P [j]] + fragSize [X [i, j]]

Fig. 3. The scripting chart of the PSO-DAP.

Table 2
Input parameters for PSO-DAP.

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 the originating 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]

number of fragments and sites [1]. Considering that other algorithms used the same approach twenty different sites or fragment numbers that are taken range from 5 to 100. Input parameters of PSO-DAP are shown in Table 2.

It is worth to note that in order to get average results, we have examined the program in 20 executions. The number of iteration is taken as 500. Also in order to compare the PSO-DAP with earlier counterparts that are presented for DAP, the application in computer with CPU 1.6 GHz, memory 4 GB and Windows 7 operating system on C# program is iterated.

Table 3 gives generation time and cost values for increasing DAP instance sizes. The obtained results of the PSO-DAP is compared with the other methods in the literature, ACO, Robust Tabu Search

Table 3
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

(RTS), GA and Hybrid Genetic Multi-Start Tabu Search Algorithm (HG-MTS) [1]. The comparison which covers cost and time values are given respectively in Tables 4 and 5. The best results are expressed in bold.

Whether or not the performance of the proposed algorithm differs from the performance of other algorithms in the literature has been determined using the sign test [28,29]. There is no significant difference between the two algorithms as H0 hypothesis. There is a significant difference between the two algorithms as H1 hypothesis. Calculations were performed at a level of five percent significance. Sign test results are detailed in Table 6. In Table 6, each algorithm has been compared with the proposed method. The statistical comparison of the proposed method and the other method is given in last six rows of the table. The H1 hypotheses were accepted because the computed Z values are outside the range in all tests ($|\text{ZValues}| > |\pm 1.96|$).

Achieved results demonstrate that PSO-DAP is less costly and time consuming than other methods (ACO, RTS, GA, HG-MTS and etc.). In this paper, DAP instances are generated randomly and the PSO-DAP is used to solve these problems. The randomly generated DAP sample set with different dimensions are located at http://www.daplib.com. To compare the PSO-DAP with the novel studies, the researchers can use these samples in future works. Due to absence of our compared algorithm's datasets, we randomly generate datasets according to formulas explained in Section 3.2. In terms of cost, among twenty exiting results, sixteen of them are the best cost and four of them (60, 65, 80 and 100 instance sizes) are close to our compared algorithm's results in Table 4. As shown in Table 6, the proposed method is statistically significantly differ-

Table 4
Cost comparison of methods for increasing DAP instance sizes (cost value is column × 10^6).

PSO-DAP (Proposed Method)		SA[9]	HG-MTS[1]	GA3[1]	GA2[1]	GA1[1]	RTS[1]	ACO[1]	Size
Standard Deviation	Average								
0.0007	**0.02**	0.04	0.04	0.04	0.04	0.04	0.04	0.04	5
0.0130	**0.05**	0.31	0.31	0.31	0.31	0.32	0.31	0.31	10
0.0532	**0.41**	0.98	0.98	0.98	0.98	0.99	0.98	0.98	15
0,0816	**0.77**	2.61	2.61	2.64	2.64	2.63	2.61	2.61	20
1.2826	**3.74**	5.15	5.19	5.24	5.26	5.25	5.19	5.19	25
1.0470	**3.19**	10.27	10.27	10.41	10.42	10.39	10.27	10.27	30
4.5046	**9.04**	16.41	16.39	16.66	16.61	16.64	16.39	16.39	35
8.9307	**19.24**	26.02	25.92	26.21	26.33	26.28	25.9	25.91	40
16.7220	**27.04**	37.40	37.27	37.82	37.8	37.73	37.26	37.28	45
25.6230	**34.43**	54.08	53.88	54.69	54.63	54.76	53.89	53.93	50
37.4836	**51.38**	71.40	71.21	72.13	72.40	72.72	71.19	71.30	55
59.9261	97.78	90.50	**90.20**	91.56	91.49	91.76	90.16	90.35	60
91.0824	125.01	112.49	**112.08**	113.84	113.75	113.59	112.13	112.31	65
109.8225	**138.69**	146.73	146.15	148.18	148.8	148.48	146.19	146.41	70
139.8139	**171.47**	178.16	177.65	180.63	180.75	180.04	177.7	177.90	75
160.2445	260.86	219.81	**219.18**	222.96	222.80	223.10	219.26	219.40	80
240.2334	**260.63**	262.89	261.99	266.19	266.15	267.04	261.88	262.24	85
302.8202	**287.09**	316.81	315.86	320.58	320.93	320.88	315.86	316.11	90
392.3170	**365.06**	371.14	369.91	375.29	375.85	375.49	369.92	370.14	95
459.9464	481.58	429.10	**427.98**	434.45	436.15	436.19	428.28	428.40	100

Table 5
Execution time (s) comparison of methods for increasing DAP instance sizes.

DAP Size	ACO [1]	RTS [1]	GA1 [1]	GA2 [1]	GA3 [1]	HG-MTS [1]	SA[9]	PSO-DAP (Proposed Method)
5	9.26	0.83	76.27	56.11	88.11	1.44	130.29	**0.74**
10	14.52	2.73	87.80	60.37	94.91	2.45	143.84	**1.35**
15	13.74	5.66	90.76	66.22	104.13	2.65	214.30	**2.17**
20	17.91	8.89	123.79	84.13	167.22	4.17	243.30	**3.65**
25	25.86	14.52	131.98	81.96	125.30	5.21	351.23	**4.25**
30	31.17	20.89	132.46	104.64	137.02	7.38	461.89	**6.45**
35	43.31	29.06	150.06	111.87	151.02	10.73	393.73	**6.81**
40	56.59	37.05	166.80	128.75	173.21	15.60	420.65	**8.85**
45	80.92	48.67	191.93	159.10	202.10	20.80	437.74	**8.42**
50	105.33	62.74	471.98	207.56	359.57	26.80	511.40	**9.60**
55	126.00	76.07	268.31	201.43	261.71	27.22	516.86	**13.74**
60	166.55	91.79	315.31	208.37	290.46	39.56	828.14	**16.09**
65	204.35	109.20	421.93	284.08	336.01	48.92	1090.77	**16.09**
70	320.62	131.54	536.15	344.20	358.03	63.13	1303.21	**17.34**
75	309.51	155.31	609.77	379.07	380.81	73.41	976.97	**17.01**
80	396.18	193.63	464.17	331.17	416.18	87.84	1234.48	**16.29**
85	807.43	195.80	532.05	364.71	586.21	102.79	898.11	**18.31**
90	621.55	215.58	563.15	400.37	531.13	123.19	1336.74	**20.98**
95	725.93	250.72	629.55	974.24	569.92	143.16	1128.08	**18.15**
100	1203.99	278.63	1236.30	568.73	808.82	179.07	1389.19	**20.97**

Table 6
Statistical comparison of the methods using sign test.

Sign	SA[9]	Sign	HG-MTS[1]	Sign	GA3[1]	Sign	GA2[1]	Sign	GA1[1]	Sign	RTS[1]	Sign	ACO[1]	PSO-DAP (Proposed Method)	Size
-	0.04	-	0.04	-	0.04	-	0.04	-	0.04	-	0.04	-	0.04	**0.02**	5
-	0.31	-	0.31	-	0.31	-	0.31	-	0.32	-	0.31	-	0.31	**0.05**	10
-	0.98	-	0.98	-	0.98	-	0.98	-	0.99	-	0.98	-	0.98	**0.41**	15
-	2.61	-	2.61	-	2.64	-	2.64	-	2.63	-	2.61	-	2.61	**0.77**	20
-	5.15	-	5.19	-	5.24	-	5.26	-	5.25	-	5.19	-	5.19	**3.74**	25
-	10.27	-	10.27	-	10.41	-	10.42	-	10.39	-	10.27	-	10.27	**3.19**	30
-	16.41	-	16.39	-	16.66	-	16.61	-	16.64	-	16.39	-	16.39	**9.04**	35
-	26.02	-	25.92	-	26.21	-	26.33	-	26.28	-	25.9	-	25.91	**19.24**	40
-	37.40	-	37.27	-	37.82	-	37.8	-	37.73	-	37.26	-	37.28	**27.04**	45
-	54.08	-	53.88	-	54.69	-	54.63	-	54.76	-	53.89	-	53.93	**34.43**	50
-	71.40	-	71.21	-	72.13	-	72.40	-	72.72	-	71.19	-	71.30	**51.38**	55
+	90.50	+	**90.20**	+	91.56	+	91.49	+	91.76	+	90.16	+	90.35	97.78	60
+	112.49	+	**112.08**	+	113.84	+	113.75	+	113.59	+	112.13	+	112.31	125.01	65
-	146.73	-	146.15	-	148.18	-	148.8	-	148.48	-	146.19	-	146.41	**138.69**	70
-	178.16	-	177.65	-	180.63	-	180.75	-	180.04	-	177.7	-	177.90	**171.47**	75
+	219.81	+	**219.18**	+	222.96	+	222.80	+	223.10	+	219.26	+	219.40	260.86	80
-	262.89	-	261.99	-	266.19	-	266.15	-	267.04	-	261.88	-	262.24	**260.63**	85
-	316.81	-	315.86	-	320.58	-	320.93	-	320.88	-	315.86	-	316.11	**287.09**	90
-	371.14	-	369.91	-	375.29	-	375.85	-	375.49	-	369.92	-	370.14	**365.06**	95
+	429.10	+	**427.98**	+	434.45	+	436.15	+	436.19	+	428.28	+	428.40	481.58	100

PSO-DAP vs. SA	PSO-DAP vs. HG-MTS	PSO-DAP vs. GA3	PSO-DAP vs. GA2	PSO-DAP vs. GA1	PSO-DAP vs. RTS	PSO-DAP vs. ACO	Statistical Notations
4	4	4	4	4	4	3	S
10	10	10	10	10	10	10	X_s
2.236	2.236	2.236	2.236	2.236	2.236	2.236	σ_s
−2.683	−2.683	−2.683	−2.683	−2.683	−2.683	−3.130	Z
±1.96	±1.96	±1.96	±1.96	±1.96	±1.96	±1.96	$Z_{\frac{0.05}{2}}$
Reject	Reject	Reject	Reject	Reject	Reject	Reject	$H0$
Accept	**Accept**	**Accept**	**Accept**	**Accept**	**Accept**	**Accept**	$H1$

ent from other methods in the literature. Consequently in terms of time consuming concept, PSO-DAP consumes short time than other compared algorithms in Table 5.

5. Conclusion

The foundation of this study is based on non-replicated data allocation in distributed database systems. The reduction of the query execution time and transaction cost values are aimed in DAP. Population-based heuristic algorithms are frequently used for carrying out this purpose. In this paper, we proposed a method based on particle swarm optimization algorithm, PSO-DAP, to minimize the query execute time and transaction cost. The performance of PSO-DAP is investigated on 20 different DAP instances, and the obtained results are compared with the results of existing methods along with considering query execution time and transaction cost. Experimental results demonstrate that the PSO-DAP is better than other compared methods in terms of solution quality and running time on almost all cases. When the dimensionality of the problem is increased, the performance of the methods is decreased because the solution space grows exponentially. But when the results are analyzed, the proposed method produces comparable results with the state-of-art algorithms. When we compared execution time of the methods, it is seen that the proposed method is superior to compared algorithms because the proposed algorithm has lower number of computation. The future works include to adaptation of the recently proposed algorithms such as firefly algorithm, bat algorithm etc. on the problems with huge size of dimensionality.

References

[1] U. Tosun, Distributed database design using evolutionary algorithms, J. Commun. Netw.-S. Kor. 16 (2014) 430–435.
[2] R.K. Adl, S.M.T.R. Rankoohi, A new ant colony optimization based algorithm for data allocation problem in distributed databases, Knowl. Inf. Syst. 20 (2009) 349–373.
[3] A. Brunstrom, S.T. Leutenegger, R. Simha, Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads, in: Proceedings of the Fourth International Conference on Information and Knowledge Management, ACM, 1995, pp. 395–402.
[4] X. Gu, W.J. Lin, B. Veeravalli, Practically realizable efficient data allocation and replication strategies for distributed databases with buffer constraints, IEEE Trans. Parall. Distr. 17 (2006) 1001–1013.
[5] W.K. Mashwani, A. Salhi, A decomposition-based hybrid multiobjective evolutionary algorithm with dynamic resource allocation, Appl. Soft Comput. 12 (2012) 2765–2780.
[6] T.C. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econ.: J. Econ. Soc. (1957) 53–76.
[7] Q. Bai, Analysis of particle swarm optimization algorithm, Comput. Inf. Sci. 3 (2010) 180–184.
[8] W. Deng, R. Chen, J. Gao, Y.J. Song, J.J. Xu, A novel parallel hybrid intelligence optimization algorithm for a function approximation problem, Comput. Math. Appl. 63 (2012) 325–336.
[9] U. Tosun, T. Dokeroglu, A. Cosar, Heuristic algorithms for fragment allocation in a distributed database system, in: Computer and Information Sciences III, Springer, 2013, pp. 401–408.
[10] U. Tosun, T. Dokeroglu, A. Cosar, A robust island parallel genetic algorithm for the quadratic assignment problem, Int. J. Prod. Res. 51 (2013) 4117–4133.
[11] N. Barbalios, P. Tzionas, A robust approach for multi-agent natural resource allocation based on stochastic optimization algorithms, Appl. Soft Comput. 18 (2014) 12–24.
[12] G. Sen, M. Krishnamoorthy, N. Rangaraj, V. Narayanan, Mathematical models and empirical analysis of a simulated annealing approach for two variants of the static data segment allocation problem, Networks 68 (2016) 4–22.
[13] M. Wang, S. Feng, C. Ouyang, Z. Li, RFID tag oriented data allocation method using artificial immune network, in: The 27th Chinese Control and Decision Conference (2015 CCDC), IEEE, 2015, pp. 5218–5223.
[14] T. Dokeroglu, Hybrid teaching-learning-based optimization algorithms for the Quadratic Assignment Problem, Comput. Ind. Eng. 85 (2015) 86–101.
[15] A. Singh, K.S. Kahlon, R.S. Virk, Replicated static allocation of fragments in distributed database design using biogeography-based optimization, in: Proc of Int. Conf. on Advances in Communication, Network, and Computing, CNC, 2014, pp. 462–472.
[16] U. Tosun, A new recombination operator for the genetic algorithm solution of the quadratic assignment problem, Procedia Comput. Sci. 32 (2014) 29–36.
[17] S.P. Li, M.H. Wong, Data allocation in scalable distributed database systems based on time series forecasting, IEEE Int. Congr. Big Data (2013) 17–24.
[18] H.I. Abdalla, A new data re-allocation model for distributed database systems, Int. J. Database Theory Appl. 5 (2012) 45–60.
[19] A.A. Amer, H.I. Abdalla, A heuristic approach to re-allocate data fragments in DDBSs, Information Technology and e-Services (ICITeS), 2012 International Conference on IEEE (2012) 1–6.
[20] Z.-J. Lee, S.-F. Su, C.-Y. Lee, Y.-S. Hung, A heuristic genetic algorithm for solving resource allocation problems, Knowl. Inf. Syst. 5 (2003) 503–511.
[21] T. Ulus, M. Uysal, Heuristic approach to dynamic data allocation in distributed database systems, Pak. J. Inf. Technol. 2 (2003) 231–239.

[22] I. Ahmad, K. Karlapalem, Y.-K. Kwok, S.-K. So, Evolutionary algorithms for allocating data in distributed database systems, Distrib. Parallel Databases 11 (2002) 5–32.
[23] J. Kennedy, R. Eberhart, Particle swarm optimization, IEEE Int. Conf. Neural Netw. Proc. 1–6 (1995) 1942–1948.
[24] M.S. Kiran, M. Gunduz, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, Appl. Soft Comput. 13 (2013) 2188–2203.
[25] M. Mahi, O.K. Baykan, H. Kodaz, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem, Appl. Soft Comput. 30 (2015) 484–490.
[26] Y. Shi, R. Eberhart, A modified particle swarm optimizer, Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on, IEEE (1998) 69–73.
[27] A.S. Mamaghani, M. Mahi, M.R. Meybodi, M.H. Moghaddam, A novel evolutionary algorithm for solving static data allocation problem in distributed database systems, Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on, IEEE (2010) 14–19.
[28] P.S. Mann, Introductory Statistics, 8th edition, Wiley, 2013.
[29] D. Lurie, L.R. Abramson, J.A. Vail, Applying Statistics, Citeseer, 2011.