Available online at www.sciencedirect.com
SCIENCE DIRECT®
Analytica Chimica Acta 509 (2004) 187–195
ANALYTICA CHIMICA ACTA
www.elsevier.com/locate/aca

An ant colony approach for clustering

P.S. Shelokar, V.K. Jayaraman, B.D. Kulkarni*

Chemical Engineering & Process Development Division, National Chemical Laboratory, Dr. Homi Bhabha Road, Pune 411 008, India

Received 2 September 2003; received in revised form 25 November 2003; accepted 11 December 2003

Abstract

This paper presents an ant colony optimization methodology for optimally clustering *N* objects into *K* clusters. The algorithm employs distributed agents which mimic the way real ants find a shortest path from their nest to food source and back. This algorithm has been implemented and tested on several simulated and real datasets. The performance of this algorithm is compared with other popular stochastic/heuristic methods viz. genetic algorithm, simulated annealing and tabu search. Our computational simulations reveal very encouraging results in terms of the quality of solution found, the average number of function evaluations and the processing time required.

Keywords: Ant colony metaheuristic; Clustering; Optimization; Euclidean distance

1. Introduction

Clustering aims to discover sensible organization of objects in a given dataset by identifying and quantifying similarities (dissimilarities) between the objects. Cluster analysis has found many applications including qualitative interpretation and data compression [1], process monitoring [2], local model development [3], analysis of chemical compounds for combinatorial chemistry [4], toxicity testing [5], finding structure–activity relations [6], discovering of clusters in DNA dinucleotides [7], classification of coals [8], etc. A good introduction to contemporary data-clustering algorithms can be found elsewhere [9]. Different criteria are employed by different authors for clustering. Most of the clustering criterion functions are nonconvex and nonlinear so that the problem may have local minimum solutions, which are not necessarily optimal [10]. Moreover, they possess exponential complexity in terms of number of clusters and become an NP-hard problem when number of clusters exceeds 3 [11]. Due to the strategic importance of clustering in many fields, several algorithms have been proposed in literature to solve clustering problems [12–17]. Recently, evolutionary and metaheuristics like, tabu search [18], genetic algorithms [14,19] and simulated annealing [20,21] have been successfully employed for clustering. In this work, we recast the recently proposed ant colony optimization algorithm to suit the need for data clustering.

Ant colony optimization (ACO) metaheuristic, a novel population-based approach was recently proposed by Dorigo et al. to solve several discrete optimization problems [22,23]. The ACO mimics the way real ants find the shortest route between a food source and their nest. The ants communicate with one another by means of pheromone trails and exchange information about which path should be followed. The more the number of ants traces a given path, the more attractive this path (trail) becomes and is followed by other ants by depositing their own pheromone. This auto catalytic and collective behavior results in the establishment of the shortest route. As shown in Fig. 1, two ants start from their nest in search of food source at the same time to different directions. One of them chooses the path that turns out to be shorter while the other takes the longer sojourn. The ant moving in the shorter path returns to the nest earlier and the pheromone deposited in this path is obviously more than what is deposited in the longer path. Other ants in the nest thus have high probability of following the shorter route. These ants also deposit their own pheromone on this path. More and more ants are soon attracted to this path and hence the optimal route from the nest to the food source and back is very quickly established. Such a pheromone-meditated cooperative search process leads to the intelligent swarm behavior. This real-life search behavior was the key motivation factor leading to the formulation of artificial ant algorithms to solve several large-scale com-

* Corresponding author. Tel.: +91-20-589-3095; fax: +91-20-589-3041.
E-mail address: bdk@ems.ncl.res.in (B.D. Kulkarni).

0003-2670/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.aca.2003.12.032

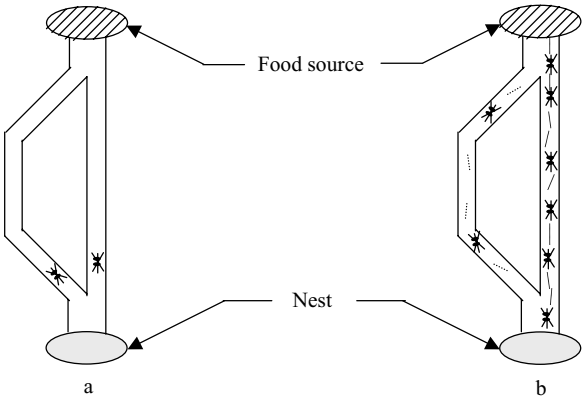

Fig. 1. Movement of ant algorithm from nest-food source and back: (a) two ants start exploring paths towards the food source and (b) pheromone is deposited more quickly on the shortest path and eventually most of the ants have chosen the shortest path.

binatorial and function optimization problems [24–26]. In all these algorithms, a set of ant-like agents or software ants solve the problem under consideration through a cooperative effort. This effort is mediated by exchanging information on the problem structure the agents concurrently collect while stochastically building solutions. Similarly, we propose an ACO algorithm for data clustering, in which a set of concurrent distributed agents collectively discover a sensible organization of objects for a given dataset.

Recently, ant-like agents have been applied to solve problems in the context of objects clustering [27–30]. In these algorithms, a population of distributed agents randomly move onto the two-dimensional grid to move objects to form clusters. Initially each agent selects a random direction among the eight possible ones. The agent has a threshold probability to further continue in the previously chosen direction when moving next, else it generates randomly a new direction. The number of moves an agent can perform is defined a priori. The agents try to pick up/drop objects on the two-dimensional board according to a local density measure of similar objects without any global control on the agents. The approach introduced in this paper is quite different from the above-mentioned ant algorithms in the context of data clustering. In our algorithm, each agent discovers a possible partition of objects in a given dataset and the level of partitioning is measured subject to some (Euclidean distance) metric. Information associated with an agent about clustering of objects is accumulated in the global information hub (pheromone trail matrix) and is used by the other agents to construct possible clustering solutions and iteratively improve them. The algorithm works for a given maximum number of iterations and the best solution found with respect to a given metric represents an optimal or near-optimal partitioning of objects into subsets in a given dataset.

This paper is organized as follows. Section 2 describes the steps involved in the ACO algorithm to solve a clustering problem, while Section 3 reports on the computational results of evaluation of the performance of the ACO algorithm on several simulated and chemical datasets. Finally, conclusions of the current work are reported in Section 4.

2. ACO algorithm for clustering problems

This section describes the ant algorithm to solve a clustering problem where the aim is to obtain optimal assignment of N objects in $\Re^n$ to one of the K clusters such that the sum of squared Euclidean distances between each object and the center of the belonging cluster is minimized. The algorithm considers R agents to build solutions. An agent starts with an empty solution string S of length N where each element of string corresponds to one of the test samples. The value assigned to an element of solution string S represents the cluster number to which the test sample is assigned in S. For example, a representative solution string, S_1 in Table 4 constructed for $N = 8$ and $K = 3$ is given below as

2	1	3	2	2	3	2	1

We note that the first element of the above string is assigned to cluster number 2, second element is allocated to cluster number 1 and so on. To construct a solution, the agent uses the pheromone trail information to allocate each element of string S to an appropriate cluster label. At the start of the algorithm, the pheromone matrix, τ is initialized to some small value, τ_0. The trail value, τ_{ij} at location (i, j) represents the pheromone concentration of sample i associated to the cluster j. For the problem of separating N samples into K clusters the pheromone matrix is of size $N \times K$. Thus, each sample is associated with K pheromone concentrations. The pheromone trail matrix evolves as we iterate. At any iteration level, each one of the agents or software ants will develop such trial solutions using the process of pheromone-mediated communication with a view to obtain a near-optimal partition of the given N test samples into K groups satisfying the defined objective. After generating a population of R trial solutions, a local search is performed to further improve fitness of these solutions. The pheromone matrix is then updated depending on the quality of solutions produced by the agents. Guided by the modified pheromone matrix, the agents build improved solutions and the above steps are repeated for certain number of iterations.

2.1. Algorithm details

As explained earlier, ants start with empty solution strings and in the first iteration the elements of the pheromone matrix are initialized to the same values. With the progress of iterations, the pheromone matrix is updated depending upon the quality of solutions produced. Let us consider for the purpose of illustration, a dataset containing $N = 8$ test samples defined by $n = 4$ attributes as shown in Table 1. The test samples are to be clustered into $K = 3$ subsets using $R = 10$ agents. We now proceed to describe the progress of

Table 1
Illustrative dataset to explain ACO algorithm for clustering with $N = 8$ and $n = 4$

N	n			
	1	2	3	4
1	5.1	3.5	1.4	0.2
2	4.9	3	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5	3.4	1.5	0.2

current iteration, t with a view to providing a clear picture of the algorithm details. The agents build their solutions by applying the information provided by the pheromone matrix updated at the end of iteration, $t - 1$. The pheromone concentrations for the first sample as shown in Table 2 are: $\tau_{11} = 0.014756$, $\tau_{12} = 0.015274$, and $\tau_{13} = 0.009900$. It indicates that at the current iteration, sample number 1 has the highest probability of belonging to cluster number 2 because τ_{12} is highest. To generate a solution S, the agent selects cluster number for each element of string S by one of the following ways:

(i) using probability q_0, cluster having the maximum pheromone concentration is chosen (q_0 being a priori defined number, $0 < q_0 < 1$, $q_0 = 0.98$ for the illustrative example and in our simulations), and/or
(ii) one of the K (three) clusters using a stochastic distribution with a probability $(1 - q_0)$, denoted as, p_{ij}.

The first process is known as exploitation whereas the latter is termed as biased exploration [31]. To explain how the above-mentioned procedures work simultaneously, consider developing first solution string, S_1 shown in Table 4. We generate numbers randomly from uniform distribution in the range between 0 and 1. The numbers generated are equal to the length of solution string. The generated random numbers are, say (0.693241, 0.791452, 0.986142, 0.988432, 0.243672, 0.967721, 0.0914324, 0.348767). Thus, elements 1, 2, 5, 6, 7, and 8 are assigned to appropriate clusters

Table 2
Pheromone trail matrix generated during run of the ACO algorithm for dataset shown in Table 1

N	K		
	1	2	3
1	0.014756	0.015274	0.009900
2	0.015274	0.009900	0.014756
3	0.015274	0.014756	0.009900
4	0.009900	0.015274	0.014756
5	0.014756	0.015274	0.009900
6	0.009900	0.014756	0.015274
7	0.009900	0.020131	0.009900
8	0.015274	0.014756	0.009900

Table 3
Normalized pheromone trail matrix

N	K		
	1	2	3
1	0.3695	0.3825	0.2479
2	0.3825	0.2479	0.3695
3	0.3825	0.3695	0.2479
4	0.2479	0.3825	0.3695
5	0.3695	0.3825	0.2479
6	0.2479	0.3695	0.3825
7	0.2479	0.5041	0.2479
8	0.3825	0.3695	0.2479

by using first procedure (i.e. clusters chosen with highest pheromone concentration referring to Table 2) since random numbers corresponding to these elements are less than q_0. On the other hand elements 3 and 4 of the solution string S_1 are assigned to one of the three clusters by using second procedure since their corresponding random numbers are higher than the threshold, q_0. The second process chooses any one of the three clusters with a normalized pheromone probability (pheromone probability normalized to 1) given by

$$p_{ij} = \frac{\tau_{ij}}{\sum_{k=1}^{K} \tau_{ik}}, \quad j = 1, \ldots, K \tag{1}$$

where p_{ij} is the normalized pheromone probability for element i belongs to cluster j. For illustration purpose the normalized pheromone matrix is shown in Table 3. For the third element of solution string S_1 the cluster number 1, 2 or 3 is selected with the normalized probabilities 0.3695, 0.3825, 0.2479, respectively. This can be readily chosen by generating a number from uniform distribution. Thus, if the random number chosen in the range (0, 1) lies between 0 and 0.3695, cluster number 1 contains the third element, if it is between 0.3695 and 0.7520, cluster number 2 is chosen and if it is greater than 0.7520 cluster number 3 is chosen for allocating the third element of S_1. Suppose the random number chosen is, say, 0.784342, which is greater than 0.7520, hence third element of string S_1 is assigned to cluster number 3. Similarly, cluster number for the other elements are assigned and the complete solution string S_1 shown in Table 4 is built. In this way, remaining nine agents can construct their solutions as given in Table 4.

The quality of solution constructed is measured in terms of the value of objective function for a given data-clustering problem. This objective function is defined as the sum of squared Euclidean distances between each object and the center of belonging cluster. Consider a given dataset of N objects $\{x_1, x_2, \ldots, x_N\}$ in $\Re^n$-dimensional space to be partitioned into a number, say K, of clusters or groups. The mathematical formulation of the data-clustering problem can be described as

$$\text{Min } F(w, m) = \sum_{j=1}^{K} \sum_{i=1}^{N} \sum_{v=1}^{n} w_{ij} ||x_{iv} - m_{jv}||^2 \tag{2}$$

Table 4
Solutions generated by $R = 10$ agents during run of the ACO algorithm

S	N								Fitness, F
	1	2	3	4	5	6	7	8	
S_1	2	1	3	2	2	3	2	1	2.695110
S_2	1	1	2	2	2	3	3	1	2.474522
S_3	1	1	1	2	2	3	2	1	1.816471
S_4	2	1	1	2	3	3	2	1	2.140193
S_5	2	2	1	2	2	3	2	1	1.982272
S_6	2	1	1	2	2	3	3	1	2.534078
S_7	2	1	1	2	2	3	2	1	1.842034
S_8	2	3	1	2	2	3	2	3	2.408086
S_9	2	1	1	2	1	3	2	1	1.900668
S_{10}	1	1	2	2	2	3	1	1	1.877386

such that

$$\sum_{j=1}^{K} w_{ij} = 1, \quad i = 1, \ldots, N \tag{3}$$

$$\sum_{i=1}^{N} w_{ij} \geq 1, \quad j = 1, \ldots, K \tag{4}$$

where x_{iv} is a value of vth attribute of ith sample; m a cluster center matrix of size $K \times n$; m_{jv} an average of the vth attribute values of all samples in the cluster j; w a weight matrix of size $N \times K$; w_{ij} an associated weight of object x_i with cluster j which can be assigned as

$$w_{ij} = \begin{cases} 1 & \text{if object } i \text{ is contained in cluster } j \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \ldots, N, \; j = 1, \ldots, K$$

Referring to first solution string S_1 in Table 4:

2	1	3	2	2	3	2	1

we note that the first element of the above string is assigned to cluster number 2 and thus $w_{11} = 0$, $w_{12} = 1$, $w_{13} = 0$. Similarly the fifth object is allocated to group number 2 and its w_{ij} vector is $w_{51} = 0$, $w_{52} = 1$, $w_{53} = 0$ and so on. After getting w_{ij}'s, the center of each cluster, m_j can be obtained as

$$m_{jv} = \frac{\sum_{i=1}^{N} w_{ij} x_{iv}}{\sum_{i=1}^{N} w_{ij}}, \quad j = 1, \ldots, K, \; v = 1, \ldots, n \tag{5}$$

For a given solution string S_1, knowing the cluster center matrix m and weight matrix w its function value can be calculated using Eq. (2). For solution S_1 (Table 4), weight matrix can be given as

N	K		
	1	2	3
1	0	1	0
2	1	0	0
3	0	0	1
4	0	1	0
5	0	1	0
6	0	0	1
7	0	1	0
8	1	0	0

Using these weights in Eq. (5), the center of each cluster m_1, m_2, and m_3 are obtained as shown below:

K	n			
	1	2	3	4
1	4.9500	3.2000	1.4500	0.2000
2	4.8250	3.4000	1.4250	0.2250
3	5.0500	3.5500	1.5000	0.3000

Substituting the weight matrix and cluster centers in Eq. (2) fitness (objective function) of solution S_1 is calculated. The computed objective function values for 10 strings are shown in Table 4.

Many of the available ACO algorithms employ some form of local search procedures with a view to improve solutions discovered by the software ants [31,32]. If heuristic information about a particular problem domain is not easily available, local search can help to find good results [33]. In these algorithms, local search procedure is applied on all the generated solutions, R or on a few percent of R. In this work, we have performed local search on L solutions representing best 20% of the total solutions. Before conducting local search, members of the population are sorted in the ascending order of their function values. A simple local search procedure is implemented on top L solutions with highest fitness values (lowest values of objective function). In our illustrative example, the sorted solution strings are shown in Table 5 and with $L = 2$ we conduct local search on top two solution strings in this table. There are various ways of conducting local search. In our work, we altered the cluster number of each sample in the solution string with certain threshold probability, p_{ls} a priori defined number in the range 0 and 1, $p_{ls} = 0.01$ for illustrative example and our simulations. Considering the topmost solution string in Table 5:

1	1	1	2	2	3	2	1

Let us first generate eight random numbers in the range between 0 and 1. Let the random number generated be (0.231345, 0.742312, 0.655361, 0.198312, 0.001636, 0.1278345, 0.874452, 0.436587). Thus, only the value

Table 5
Solutions in Table 4 sorted as per the criterion of clustering problem

S	N								Fitness, F
	1	2	3	4	5	6	7	8	
S_1	1	1	1	2	2	3	2	1	1.816471
S_2	2	1	1	2	2	3	2	1	1.842034
S_3	1	1	2	2	2	3	1	1	1.877386
S_4	2	1	1	2	1	3	2	1	1.900668
S_5	2	2	1	2	2	3	2	1	1.982272
S_6	2	1	1	2	3	3	2	1	2.140193
S_7	2	3	1	2	2	3	2	3	2.408086
S_8	1	1	2	2	2	3	3	1	2.474522
S_9	2	1	1	2	2	3	3	1	2.534078
S_{10}	2	1	3	2	2	3	2	1	2.695110

of random number corresponding to fifth element is less than the threshold probability, 0.01. So only, this element has to be assigned a different cluster number. Currently the fifth element is assigned to cluster number 2. Therefore, it has to be assigned to either cluster number 1 or 3 with equal probability by generating a random number. The solution string (LS_1) obtained in neighborhood of topmost solution string S_1 by local search is given below as

LS_1	1	1	1	2	1	3	2	1

It can be observed that the fifth element has been relocated to cluster number 1. Similarly, the second string, S_2 from top in Table 5 undergoes local search operation and the solution string (LS_2) generated in neighborhood of S_2 is given as

LS_2	2	1	1	2	2	3	1	1

After conducting the local search, the objective function values for the newly generated solutions are computed using Eq. (2). These solutions can be accepted only if there is improvement in the fitness. The objective function values for locally generated solutions, LS_1 and LS_2 are 1.593560 and 1.835535, respectively. Here, the quality of both solutions generated is better than that of solutions S_1 and S_2 (1.816471 and 1.842034, respectively) given in Table 5. Therefore, newly generated solutions replace these two solutions in Table 5. The local search algorithm can be written as follows:

With local search probability threshold p_{ls} in $[0, 1]$, a neighbor of S_k, $k = 1, \ldots, L$ is generated as

(i) $k = 1$.
(ii) Let S_t be a temporary solution and assign $S_t(i) = S_k(i)$, $i = 1, \ldots, N$.
(iii) For each element i of S_t, draw a random number r in $(0, 1)$. If $r \leq p_{ls}$, an integer j in the range $(1, K)$, such that $S_k(i) \neq j$ is randomly selected and let $S_t(i) = j$.
(iv) Calculate cluster centers and weights associated with solution string S_t and find its objective function value using (2) as F_t. If F_t is less than F_k, then $S_k = S_t$ and $F_k = F_t$.
(v) $k = k + 1$; if $k \leq L$ go to step (ii), else stop.

After performing the local search operation, the pheromone matrix is updated. Such a pheromone updating process reflects the usefulness of dynamic information provided by the software ants. Thus, the pheromone matrix is a kind of adaptive memory that contains information provided by the previously found superior solutions, and is updated at the end of iteration. The trail updating process applied in this algorithm considers best L solutions out of R members discovered by the agents as per the given criterion (Eq. (2)) at iteration level t. These L agents mimic deposition of the pheromone trail of real ants by assigning some real numbers τ_{ij} associated with solution attributes. The trail information is updated using the following rule as

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \sum_{l=1}^{L} \Delta\tau_{ij}^{l}, \quad i = 1, \ldots, N,\ j = 1, \ldots, K \tag{6}$$

where ρ is the persistence of trail that lies between $[0, 1]$ and $(1 - \rho)$ the evaporation rate. Higher value of ρ suggests that the information gathered in the past iterations is forgotten faster. The amount $\Delta\tau_{ij}^{l}$ is equal to $1/F_l$, if cluster j is assigned to ith element of the solution constructed by ant l and zero otherwise. An optimal solution is that solution which minimizes the objective function value (2). The value of best solution in memory is updated with the value of the solution obtained as "current iteration best solution" if it is having a lower objective function value than that of the best solution in memory. This comprises one iteration of the algorithm.

Thus, at any iteration level the algorithm essentially executes three steps viz. (1) generation of new R solutions by software ants using the modified pheromone trail information available from previous iteration, (2) performing local search operation on the newly generated solutions, and (3) updating pheromone trail matrix. The algorithm repeatedly carries out these three steps for a maximum number of given iterations, and solution having lowest function value represents the optimal partitioning of objects of a given dataset into several groups. The summary of ant algorithm for data clustering is depicted as a flowchart shown in Fig. 2.

3. Results and discussion

We implemented the ACO clustering algorithm on five datasets. All algorithms are executed in C++ language and all experiments are performed on a Pentium IV 400 MHz Personal Computer. The five datasets (two simulated and three chemical datasets) are described below.

Both simulated datasets were created using a random number generator that produced Gaussian distributed set of

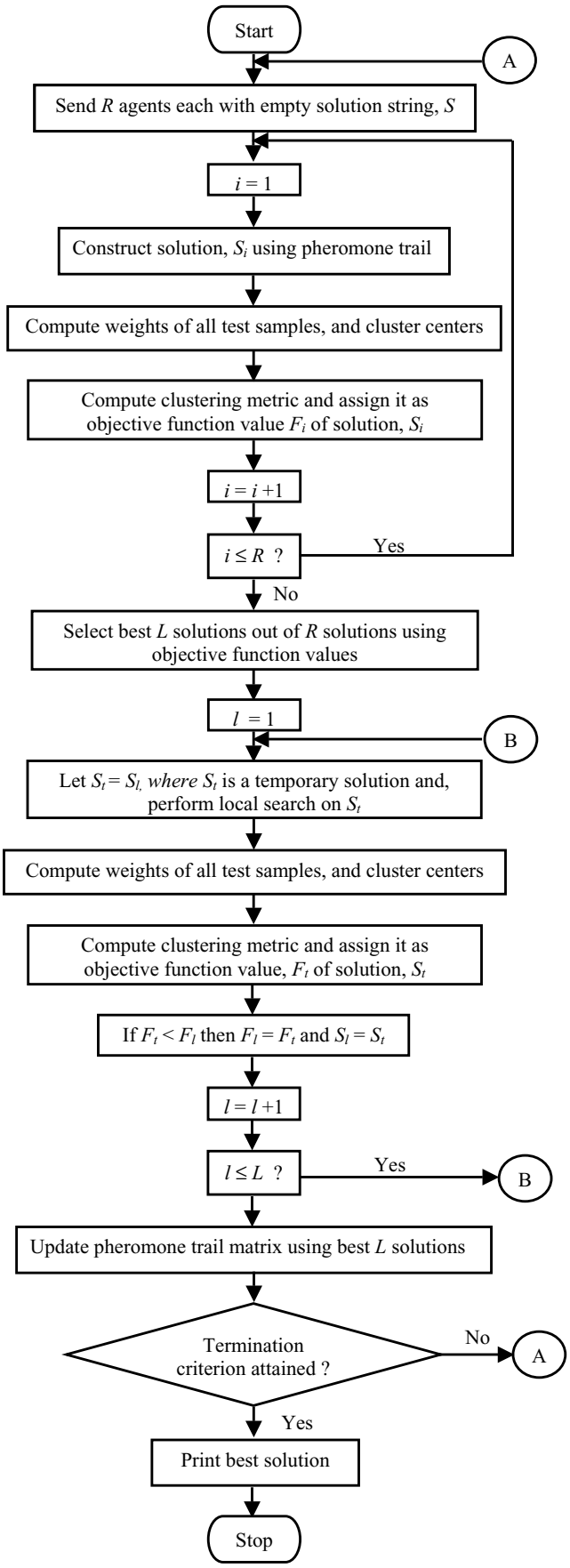

Fig. 2. Flowchart of ant algorithm for data clustering.

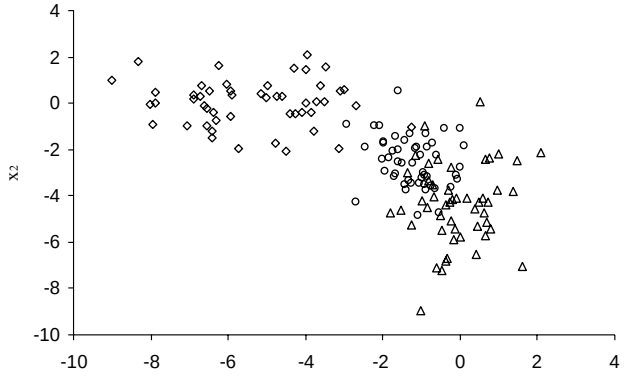

Fig. 3. Example 1: (◇) objects from class 1; (△) objects from class 2; (○) objects from class 3.

objects. These datasets are given as

- *Example 1*. This dataset is composed of $K = 3$ clusters with 50 objects in each cluster. The data was generated using mean $\mu_1 = [3, 0]$, $\mu_2 = [0, 3]$, $\mu_3 = [1.5, 2.5]$ and variance $\lambda_1 = [0.3, 1]$, $\lambda_2 = [1, 0.5]$, $\lambda_3 = [2, 1]$. The dataset is shown in Fig. 3.
- *Example 2*. The dataset represents $K = 6$ clusters with allocation of 25 objects to each cluster. The data was simulated using mean values $\mu_1 = [3, 0]$, $\mu_2 = [0, 3]$, $\mu_3 = [1.5, 2.5]$, $\mu_4 = [0.2, 0.1]$, $\mu_5 = [1.2, 0.8]$, $\mu_6 = [0.1, 1.1]$ and variance $\lambda_1 = [0.3, 1]$, $\lambda_2 = [1, 0.5]$, $\lambda_3 = [2, 1]$, $\lambda_4 = [0.03, 1]$, $\lambda_5 = [2, 0.5]$, $\lambda_6 = [0.2, 0.4]$. The dataset is given in Fig. 4.

Many authors have considered the *iris*, *wine* and *human thyroid disease* datasets as data-clustering problems to study and evaluate the performance of their algorithms. These are briefly described as

- *Example 3*. The dataset consists of $N = 150$ samples of three iris flowers ($K = 3$) viz. *setosa*, *versicolor*, and *verginica*. Each object is defined by four attributes, $n = 4$: sepal length, sepal width, petal length, and petal width. The data is obtained from the UCI repository of machine learning databases [34].
- *Example 4*. This dataset contains chemical analysis of $N = 178$ wines, derived from three different cultivars, $K = 3$. Wine type is based on 13 continuous attributes, $n = 13$ derived from chemical analysis: alcohol, malic acid, ash, alcalinity of ash, magnesium, total phenols, flavanoids, nonflavanoids phenols, proanthocyaninsm, color intensity, hue, OD280/OD315 of diluted wines and proline. It is also available in the public domain of UCI repository of machine learning databases [34].
- *Example 5*. This dataset categories $N = 215$ samples of patients suffering from three human thyroid diseases, $K = 3$ as: euthyroid, hyperthyroidism, and hypothyroidism patients where 150 individuals are tested euthyroid thyroid, 30 patients are experienced hyperthyroidism thyroid while 35 patients are suffered by hypothyroidism thyroid. Each individual was characterized by the result of five, $n =$

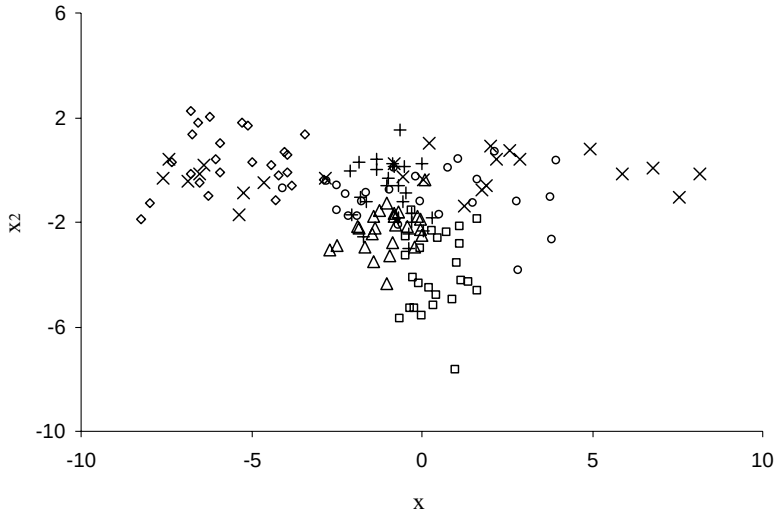

Fig. 4. Example 2: (◇) objects from class 1; (□) objects from class 2; (△) objects from class 3; (×) objects from class 4; (+) objects from class 5; (○) objects from class 6.

Table 6
Results obtained by the four algorithms for 10 different runs on Example 1

Method	Function value			Function evaluations	CPU time (s)
	F_{best}	F_{avg}	F_{worst}		
ACO	203.595559	203.626619	203.906163	12396	31.49
GA	203.595559	204.057260	204.689421	32757	69.89
TS	204.053636	204.436562	205.381524	23401	61.69
SA	203.595559	203.706785	203.897976	27505	72.93

5 laboratory tests as: total serum thyroxine, total serum tri-iodothyronine, serum tri-iodothyronine resin uptake, serum thyroid-stimulating hormone (TSH), and increase TSH after injection of TSH-releasing hormone [35].

To evaluate the performance of the ACO algorithm, we have compared it with several typical stochastic algorithms including the simulated annealing (SA) approach [20], the genetic algorithms (GA) [19] and the tabu search (TS) approach [18]. The effectiveness of stochastic algorithms is greatly dependent on the generation of initial solutions. Therefore, for every dataset, algorithms performed 10 times individually for their own effectiveness tests, each time with randomly generated initial solutions. Each experiment is made of at most 1000 iterations of the associated search procedure by the ACO algorithm, the GA approach and the TS algorithm. For each test, the SA procedure was called at most 30 000 times. The comparison of results for each dataset is based on the best solution found in 10 distinct runs of each algorithm, the average number of function evaluations required and the average processing time taken to attain the best solution. The solution quality is also given in terms of the average and worst values of the clustering metric (F_{avg}, F_{worst}, respectively) after 10 different runs for each of the four algorithms. For clustering problem, Example 1 results given in Table 6, show that the ACO, GA and SA clustering algorithms provide the optimum value of 203.595559. In fact, the ACO found this optimum nine times as compared to the five times and one time obtained by the SA and GA approach in 10 runs, respectively. The average number of function evaluations to obtain the best solution and the average time required to attain the convergence are 12 396 and 31.49, respectively, for the ACO algorithm, which are better than the other algorithms as shown

Table 7
Results obtained by the four algorithms for 10 different runs on Example 2

Method	Function value			Function evaluations	CPU time (s)
	F_{best}	F_{avg}	F_{worst}		
ACO	172.948099	173.364862	173.613300	25260	66.21
GA	173.990484	177.266506	185.867600	40065	93.79
TS	176.576398	178.870536	180.802676	28191	83.91
SA	173.244913	174.572357	177.778584	30000	79.48

Table 8
Results obtained by the four algorithms for 10 different runs on Example 3

Method	Function value			Function evaluations	CPU time (s)
	F_{best}	F_{avg}	F_{worst}		
ACO	97.100777	97.171546	97.808466	10998	33.72
GA	113.986503	125.197025	139.778272	38128	105.53
TS	97.365977	97.868008	98.569485	20201	72.86
SA	97.100777	97.136425	97.263845	29103	95.92

Table 9
Results obtained by the four algorithms for 10 different runs on Example 4

Method	Function value			Function evaluations	CPU time (s)
	F_{best}	F_{avg}	F_{worst}		
ACO	16530.533807	16530.533807	16530.533807	9,306	68.29
GA	16530.533807	16530.533807	16530.533807	33,551	226.68
TS	16666.226987	16785.459275	16837.535670	22,716	161.45
SA	16530.533807	16530.533807	16530.533807	7,917	57.28

Table 10
Results obtained by the four algorithms for 10 different runs on Example 5

Method	Function value			Function evaluations	CPU time (s)
	F_{best}	F_{avg}	F_{worst}		
ACO	10111.827759	10112.126903	10114.819200	25626	102.15
GA	10116.294861	10128.823145	10148.389608	45003	153.24
TS	10249.72917	10354.315021	10438.780449	29191	114.01
SA	10111.827759	10114.045265	10118.934358	28675	108.22

in Table 6. For clustering problem, Example 2 the ACO and SA approach provide the optimum value of 172.984099. From Table 7, the F_{avg} of 173.364862 obtained by the ACO algorithm is less than the best solution obtained by the GA and TS approach. In terms of the number of function evaluations and the processing time required, the ACO algorithm fairs better than its counterparts.

The *iris* dataset is Example 3. It contains 150 objects to be partitioned into three clusters. For this problem, the ACO and SA methods obtain the best solution of 97.100777. The ACO was able to find the optimum nine times as compared to that of five times obtained by the SA. Table 8 shows that ACO required the least number of function evaluations (10 998) and the processing time (33.72). The results obtained for the clustering problem, Example 4 are given in Table 9. The ACO, SA and GA approach provide the optimum solution of 16530.533807. The ACO, SA and GA methods found this optimum solution in all their 10 runs. The function evaluations and the execution time taken by the ACO algorithm are higher than the SA approach but less than that of the GA and TS approaches. The *human thyroid disease* dataset (Example 5) consist of 215 objects to be allocated to three clusters. Both the ACO and SA algorithms provide the optimum solution of 10111.827759 to this problem with success rate of 90 and 30% during 10 runs, respectively. In terms of the function evaluations and the process-

Table 11
Values of parameters of each of the four algorithms

ACO		GA		TS		SA	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Ants (R)	50	Population size	50	Tabu list size	25	Probability threshold	0.98
Probability threshold for maximum trail (q_0)	0.98	Crossover rate	0.8	Number of trial solutions	40	Initial temperature	5
Local search probability (p_{ls})	0.01	Mutation rate	0.001	Probability threshold	0.98	Temperature multiplier	0.98
Evaporation rate (ρ)	0.01	Maximum number of iterations	1000	Maximum number of iterations	1000	Final temperature	0.01
Maximum number of iterations (*itermax*)	1000					Number of iterations detect steady state	100
						Maximum number of iterations	30000

ing time, the ACO performed better than the SA, GA and TS clustering algorithms as can be observed from Table 10.

Several simulations were performed to find the algorithmic parameters that result into the best performance of all the algorithms in terms of the quality of solution found, the function evaluations and the processing time required. The algorithmic parameters used in this study are given in Table 11.

In this study, several datasets were considered with clusters ranging from $K = 3$ to $K = 6$ and number of attributes from $n = 2$ to $n = 13$. As seen, the results obtained by the ACO method are superior to that of the SA, GA and TS techniques. The results illustrate that the proposed ant colony optimization approach can be considered as a viable and an efficient heuristic to find optimal or near-optimal solutions to clustering problems of allocating N objects to K clusters.

4. Conclusions

In summary, an ant colony optimization algorithm to solve clustering problems has been developed in this paper. The software ants use pheromone matrix a kind of adaptive memory, which guide other ants towards the optimal clustering solution. The pheromone (weight) deposition at location (i, j) (i.e. allocation of sample i to the cluster j in a constructed solution) depends on its objective function value (smaller function value deposit higher pheromone) and the evaporation rate. The evaporation rate is a kind of forgetting factor that helps to look into other clustering locations of object i. Therefore, it will surely provide an optimal cluster representation for a clustering problem as iterations progress.

The ACO algorithm for data clustering can be applied when the number of clusters is known a priori and are crisp in nature. To evaluate the performance of the ACO algorithm, it is compared with other stochastic algorithms viz. genetic algorithm, simulated annealing and tabu search. The algorithm has been implemented and tested on several simulated and real datasets; preliminary computational experience is very encouraging in terms of the quality of solution found, the average number of function evaluations and the processing time required.

Acknowledgements

Financial support received from the Department of Science and Technology, New Delhi, India is gratefully acknowledged. The author PS thanks the Council of Scientific and Industrial Research (CSIR), the Government of India, New Delhi, for a Senior Research Fellowship.

References

[1] K.J. Mo, S. Eo, D. Shin, E.S. Yoon, Comput. Chem. Eng. 22 (1998) 555–562.
[2] P. Teppola, S.-P. Mujunen, P. Minkkinen, Chemometr. Intell. Lab. Syst. 45 (1999) 23–38.
[3] M. Ronen, Y. Shabtai, H. Guterman, Biotech. Bioeng. 77 (2002) 420–429.
[4] A. Linusson, S. Wold, B. Nordén, Chemometr. Intell. Lab. Syst. 44 (1998) 213–227.
[5] R.G. Lawson, P.C. Jurs, J. Chem. Inf. Comput. Sci. 30 (1990) 137–144.
[6] W.J. Dunn, M.J. Greenberg, S.S. Callejas, J. Med. Chem. 19 (1976) 1299–1301.
[7] M.L.M. Beckers, W.J. Melssen, L.M.C. Buydens, Comput. Chem. 21 (1997) 377–390.
[8] L. Kaufman, A. Pierreux, P. Rousseuw, M.P. Derde, M.R. Detaevernier, D.L. Massart, G. Platbrood, Anal. Chim. Acta 153 (1983) 257–260.
[9] J.W. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, San Francisco, CA, 2001.
[10] S.Z. Selim, M.A. Ismail, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 81–87.
[11] J.W. Welch, J. Stat. Comput. Simulat. 15 (1983) 17–25.
[12] D. Fisher, Mach. Learn. 2 (1987) 139–172.
[13] J. Banfield, A. Raftery, Biometrics 49 (1993) 803–821.
[14] J.-H. Jiang, J.H. Wang, X. Chu, R.-Q. Yu, Anal. Chim. Acta 354 (1997) 263–274.
[15] K. Szczubialka, J. Verdú-Andrés, D.L. Massart, Chemometr. Intell. Lab. Syst. 41 (1998) 145–160.
[16] J.A. Fernández Pierna, D.L. Massart, Anal. Chim. Acta 408 (2000) 13–20.
[17] T.N. Tran, R. Wehrens, L.M.C. Buydens, Anal. Chim. Acta 490 (2003) 303–312.
[18] K.S. Al-Sultan, Pattern Recogn. 28 (1995) 1443–1451.
[19] C.A. Murthy, N. Chowdhury, Pattern Recogn. Lett. 17 (1996) 825–832.
[20] S.Z. Selim, K.S. Al-Sultan, Pattern Recogn. 24 (1991) 1003–1008.
[21] L.-X. Sun, Y.-L. Xie, X.-H. Song, J.-H. Wang, R.-Q. Yu, Comput. Chem. 18 (1994) 103–108.
[22] M. Dorigo, V. Maniezzo, A. Colorni, IEEE Trans. Syst. Man Cybern. 26 (1996) 29–41.
[23] M. Dorigo, G. Di Caro, L.M. Gambardella, Artif. Life 5 (1999) 137–172.
[24] D. Costa, A. Hertz, J. Operat. Res. Soc. 48 (1997) 295–303.
[25] G. Di Caro, M. Dorigo, J. Artif. Intell. Res. 9 (1998) 317–365.
[26] R. Schoonderwoerd, O. Halland, J. Bruten, L. Rothkrantz, Adapt. Behav. 5 (1996) 169–207.
[27] J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, L. Chretien, *From Animals to Animats1*, in: J.A. Meyer et, S.W. Wilson (Eds.), MIT Press, Cambridge, MA, 1991, pp. 356–363.
[28] E.D. Lumer, B. Faieta, *From Animals to Animats3*, in: D. Cliff, P. Husbands, J.A. Meyer, W. Stewart (Eds.), MIT Press, Cambridge, MA, 1994, pp. 501–508.
[29] P. Kuntz, P. Layzell, D. Snyers, J. Heuristics 5 (1998) 327–351.
[30] N. Monmarché, M. Slimane, G. Venturini, in: D. Floreano, J.D. Nicoud, F. Mondala (Eds.), Lecture Notes in Artificial Intelligence, Springer-Verlag, 1999, pp. 626–635.
[31] L.M. Gambardella, M. Dorigo, INFORMS J. Comput. 12 (2000) 237–255.
[32] L.M. Gambardella, É.D. Taillard, G. Agazzi, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 63–76.
[33] T. Stützle, H. Hoos, Proceedings of the Second International Conference on Metaheuristics, Sophia-Antipolis, France, July 21–24, 1997, pp. 309–314.
[34] UCI Repository of Machine Learning Databases retrieved from the World Wide Web: http://www.ics.uci.edu/~mlearn/MLRepository.html.
[35] D. Coomans, M. Jonckheer, D.L. Massart, I. Broechaert, P. Blockx, Anal. Chim. Acta 103 (1978) 409–415.