International Journal of Management Science
Vol 15, No 1, May 2009

A Stigmergy-and-Neighborhood Based Ant Algorithm for Clustering Data*

Heesang Lee**
Department of Systems Management Engineering
Sungkyunkwan University, Suwon, Gyeonggi-do, 440-746, Korea

Gyuseok Shim
Department of Systems Management Engineering
Sungkyunkwan University, Suwon, Gyeonggi-do, 440-746, Korea

(Received: October 12, 2008 / Accepted: October 21, 2008)

ABSTRACT

Data mining, specially clustering is one of exciting research areas for ant based algorithms. Ant clustering algorithm, however, has many difficulties for resolving practical situations in clustering. We propose a new grid-based ant colony algorithm for clustering of data. The previous ant based clustering algorithms usually tried to find the clusters during picking up or dropping down process of the items of ants using some stigmergy information. In our ant clustering algorithm we try to make the ants reflect neighborhood information within the storage nests. We use two ant classes, search ants and labor ants. In the initial step of the proposed algorithm, the search ants try to guide the characteristics of the storage nests. Then the labor ants try to classify the items using the guide information that has set by the search ants and the stigmergy information that has set by other labor ants. In this procedure the clustering decision of ants is quickly guided and keeping out of from the stagnated process. We experimented and compared our algorithm with other known algorithms for the known and statistically-made data. From these experiments we prove that the suggested ant mining algorithm found the clusters quickly and effectively comparing with a known ant clustering algorithm.

Keywords: Ant Mininning, Stigmergy, Neighborhood Information, Data Clustering

---

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2005-041-D00885) and by the Brain Korea 21 Project in 2007.

** Corresponding author, E- mail: leehee@skku.edu

1. Introduction

The clustering of the data sets becomes necessary and meaningful because of rapid evolution in technology and business environment. Various clustering methods have been studied and applied to practical cases for many years. Well-known clustering methods are K-means clustering, CART (Classification And Regression Trees), SOM (Self-Organizing Map). The clustering methods can be classified into four main classes; *partitioning method, hierarchical method, density-based clustering* and *grid-based clustering*.

Ant-based clustering falls into the grid-based clustering method class. It models some behavior observed in real ants, which considers data as corps to carry out the clustering by mimicking picking up process and dropping process of ants in corpses piling. It is an instance of the broad category of ant algorithm [3], and a branch of ant colony optimization [1] or swarm intelligence [2]. The ant-based clustering is excellent in visualization due to its capability of forming a self-organizational cluster. To date, it has been studied and applied to practical cases actively, such as text mining in document.

In this paper, we propose a new grid-based ant colony algorithm, search-ant and labor-ant clustering (SLAC) algorithm for clustering of data items utilizing the behavior of ants not only in corpse filing but also in food searching and storing. We are using two kinds of ants: search ants and labor ants. At an early phase of the algorithm, a fixed number of storage nests which are storage places for data items are formed. Search ants form the characteristics of the storage nests by gathering items in early stage and labor ants are traveling to the storage nests with picked items. With this storage nest set up and using two kinds of ants, we expect that the performance of our algorithm is faster and more efficient in clustering than usual ant mining algorithms. We can also reduce and simplify the traveling paths of ants by forming a fixed storage nest.

The remainder of this paper is structured as follows. Section 2 introduces ant-based clustering briefly. Section 3 discusses our proposed algorithm. In section 4, we evaluate the proposed algorithm by computational simulations. The conclusion is followed in section 5.

2. Ant-based clustering

In 1990, Deneubourg *et al.* [4] developed the first ant clustering algorithm based on mimicking corpse piling process of real ants. In his algorithm, agents pick up and drop the data items based on the similarity of agents' local neighborhoods. Lumer and Faieta [5] improved Deneubourg's algorithm with measuring dissimilarity formula and updating the short term memory. Kunz *et al.* [6] modified Lumer and Faieta's algorithm to apply graph partitioning and other related work. Handle *et al.* [7] applied Lumer and Faieta's algorithm to text mining and introduced a number of modifications to improve; modified threshold, modified neighborhood function, direct jump, and activity-based α-adaptation [8, 9, 10].

New approaches of ant clustering algorithms which are not using Deneubourg's corpse piling model have also been studied. Labroche *et al.* [11, 12] developed a new algorithm, AntClust, based on chemical odor and some behavioral rules.

3. Search ant and Labor ant Clustering Algorithm

3.1 Main ideas

Reviewing the movement of ant and data in existing ant-based clustering methods [4, 5, 6, 7, 8, 9, 10], we can notice the following phenomena. Firstly, the data items are merged into a big cluster then moved along a square lattice. The big mass is divided into several partitioned clusters later as the clustering carried out. Several small clusters can be merged into a big cluster or can be divided into smaller clusters further. Figure 1 shows a typical progress of the current merge-and-divide type ant clustering. (We implement ATTA algorithm in [10]). The data items scattered randomly on the lattice merged into a big mass then divided later on. The above and below cluster of final screen capture is a one cluster connected each other because the algorithm uses a torus space. Ants are having longer traveling path under this process and sometimes clustering is taking too long for a large volume of data set. These algorithms do not set in prior the number of clusters. It is also hard to decide when the clustering is completed since the number of clusters can be changed some times when the number

of iterations is changed. It also shows a relatively ambiguous boundary for distinguish the clusters.

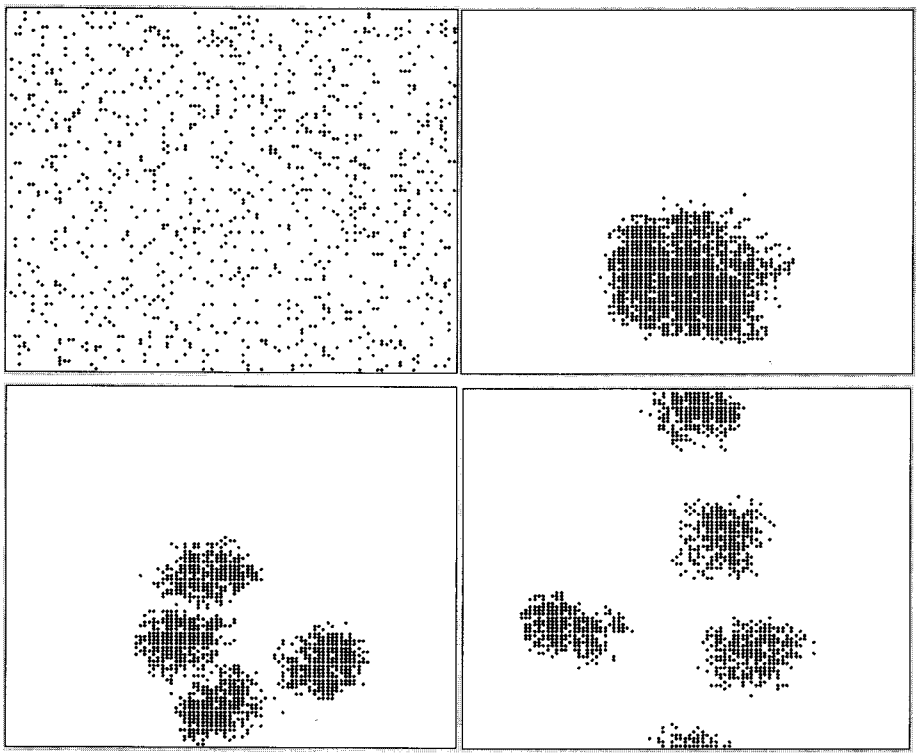


Figure 1. Typical Progress of ATTA Algorithm

In our proposed SLAC algorithm, the number of clusters should be provided by users. By doing so, our algorithm has some advantage of applications since users set the number of clusters in prior at many applications. Usual ant clustering algorithm sometimes give different number of clusters by performing different number of iteration, while SLAC algorithm does not have this evasiveness.

We set a central point of each storage nest on the lattice to form a storage nest around the central point. The number of storage nest is the number of clusters fixed by the user. A search ant drops the data items around the central point of a storage nest. The data items in the same storage nest will be similar to each other, but dissimilar to the data items of other storage nest. The data dropped by search ants is stored in a *nest memory*. A stored data item in the nest memory will not be moved later on

but keep to stay in the storage nest. Hence the nest memory is pheromone information for the search ants and the labor ants for collecting and clustering data items. The number of stored data items is decided by users and assigned equally to each storage nest. The role of search ants in initial phase terminated after this pre-defined number of data items are assigned to each storage nest. Figure 2 shows a typical progress of SLAC with the same data set of Figure1. The first screen capture shows four central points of storage nest. The second screen shows the date items placed by search ants around the storage nest, and little works has been done by labor ants in this stage. The third screen shows forming clusters each of those has dense data items dropped by labor ants. The last screen provides the final clusters established by labor ants.

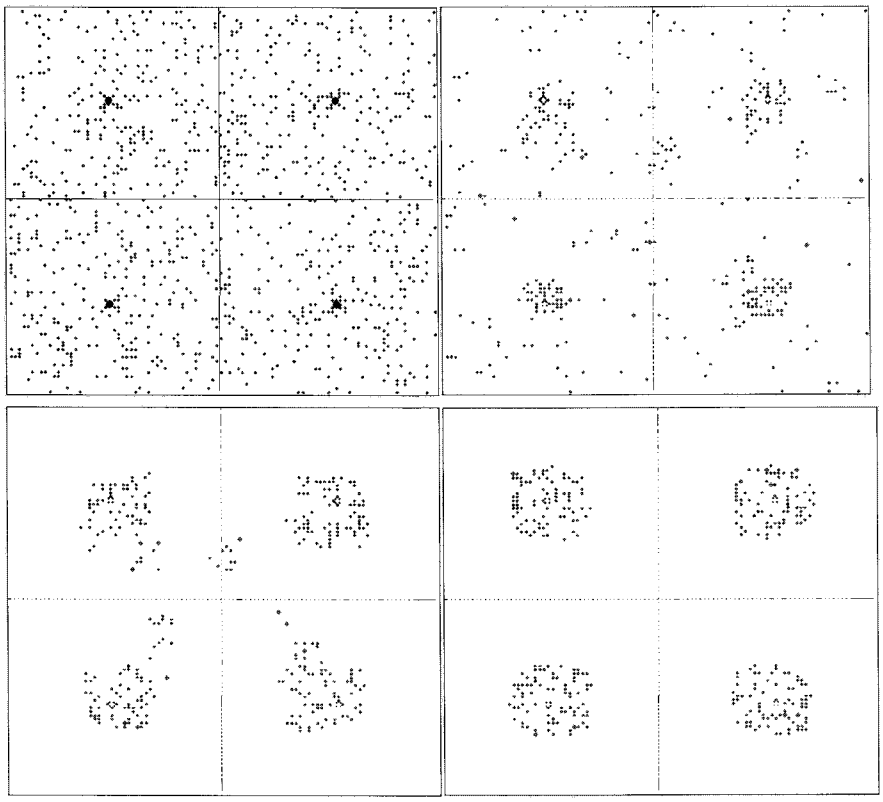


Figure 2. Typical Progress of SLAC Algorithm

In SLAC, we use Lumer and Faieta's neighborhood function in [5] for dissimilarity measure and Deneubourg's threshold function in [4] for pick up and drop prob-

abilities of the items. Lumer and Faieta in [5] suggested a *neighborhood function f(i)* (measuring dissimilarity formula) as follows:

$$f(i) = \max(0, \frac{1}{\sigma^2}\sum_{j \in L}\left(1 - \frac{\delta(i, j)}{\alpha}\right) \quad (1)$$

, where i is a data item in current site, j is an item in local neighborhood L, and $\delta(i, j) \in [0, 1]$ is a dissimilarity function. $\alpha \in [0, 1]$ is a data-dependent scaling parameter, σ^2 the diameter of L.

Deneubourg's model proposed discrete dynamics in square lattice occupied by ants and data items. At each move time step, ants move randomly from the present position to one of the four neighboring sites (left, right, above or below). If an unloaded ant encounters an item while traveling, picks the item up with the pick up probability. If an ant picks up an item, random moves to neighboring sites and try to drop the item with the drop probability. Deneubourg [4] suggested a pick-up probability and a drop probability at each site by using the *threshold functions* (2) and (3), respectively.

$$p_p(i) = \left(\frac{k^+}{k^+ + f(i)}\right)^2 \quad (2)$$

$$p_d(i) = \left(\frac{f(i)}{k^- + f(i)}\right)^2 \quad (3)$$

, where k^+, k^- are parameters (commonly $k^+ = 0.1$ and $k^- = 0.3$).

This neighborhood function, the pick up probability increases as the number of dissimilar items is high in a neighborhood, and the drop probability increases as the number of similar items is high in the neighborhood.

After the role of search ants is completed, labor ants pick up the data items using the probability by the formula (2). A labor ant with a data item moves a step toward the most similar storage nest after inspecting the similarity that is calculated by the formula (1) with the data items moved by the search ants. At each step, the data dropping decision will be made based on the drop probability calculated by the formula (3).

3.2 Main Algorithm

The main algorithm of SLAC is introduced in Algorithm 1. Data items are initially scattered on the lattice randomly. A number of the storage nests are formed by the search ants. All the labor ants move toward the randomly selected locations where data items are located to pick up a data item. The labor ants start their picking up and dropping jobs. The algorithm terminates once their prefixed number of jobs are finished.

- **Algorithm 1: Main Algorithm for SLAC**

  procedure *Main*
  //Activate search ant
  call *ActiveSearchAnt*
  for all agents **do**
  randomly select a data item
  move to the item
  pick up the item
  end for
  for all iteration **do**
  //Activate labor ant
  call *ActiveLaborAnt*
  end for
  end procedure

3.3 Search Ant Procedure

Each cell of the storage nest will be filled with the number of data items dropped by the search ants. We prepare a *nest memory* for the storage nests. The attributes of each data item are stored in the nest memory of that storage nest. The number of data item which are picked up by the search ants and moved to the storage nests is provided by the user.

At the initial phase, a search ant drops a picked data into a random storage nest. The dropped data item in the storage nest is stored in the nest memory. A search ant picks up a data item randomly and calculates the values of neighborhood functions for each data items for the storage nests from formula (1), then decides to move a storage nest that has the most data items. If the nest memory of the most data items

has remaining cells of the nest memory (i.e. the data items to be filled up by the search ants for the storage nest is remaining), the search ants check the possibility of dropping the data item based on the probability from formula (3). In case of drop decision success, drop the data item in this storage nest, and store in the corresponding nest memory. For the case of a drop decision fail or the nest memory for the storage nest is full, check if there is an empty storage nest. If there is an empty storage nest, before moving to the empty storage nest, calculate the pick up probability of the current carrying data item with the data items of the current nest memory of the current storage nest, based on the pick up probability of formula (2). This is a conservative recheck for the current carrying item for the correct membership of the clusters. If the pick up decision fails then the carrying item is dropped in the original place. If the picking up decision is success, then move to the empty storage nest and drop the item in the empty storage nest. The search iterations are terminated if such an empty storage nest does not exist.

- **Algorithm 2: Search Ant Procedure**

  procedure *ActiveSearchAnt*
  //initialize Nest memory
  for all clusters **do**
  initialize the nest and its memory
  end for
  //initial selection of random data
  randomly select an item and pick it up
  move to an empty nest and drop the item
  memorize the item and the position to nest memory
  //filling the nest
  while all nest memory is not full **do**
  randomly pick up an item
  get the best non empty matched nest
  move to the storage nest
  if the nest memory is not full
  drop the item with drop probability
  end if
  if drop fails

```
      if empty nest exists
        get pick up probability with best matched nest (ie, current nest)
       if pick up success
          move to the an empty nest and drop the item
       end if
      end if
  end if
  if all dropping trials are failed
    restore the item to the original place
  end if
  end while
  end procedure
```

3.4 Labor Ant Procedure

The data items are going to be dropped by the labor ants around the central points of the storage nests using the information regarding the items that have been set by the search ants in each storage nest.

We select a labor ant randomly then set the next direction of move according to the highest value of neighborhood function for every storage nest using the formula (1). A labor ant moves toward that direction in step size, then it makes the dropping decision according to the drop probability of formula (3). We then terminate the current iteration if the labor ants failed to drop. Otherwise we continue until an arbitrary free data is picked using pick up probability of formula (2)

- **Algorithm 3: Labor Ant Procedure**

```
procedure ActiveLaborAnt
randomly select an agent
step to the best matched nest with step size
get the drop probability
drop its item with probability
if dropping is not failed
    while the agent has no item do
        randomly select an item
        get the pick up probability
```

pick up the item with pick-up probability
end while
end if
end procedure

In our SLAC algorithm, cluster is identified by the membership of data items by calculating the distances between the center points of the storage nests and location of each data items. In SLAC the identification of the clusters is distinct since we also set the center points sufficiently separated each other in the lattice and the ants try to make clusters be formed around the center points.

4. Experiments and results

4.1 Experimental design

In our experiment, a real data set and synthetic data sets generated by two-dimensional Normal distribution $N(\mu, \sigma)$ are used. The synthetic data sets are generated as in Table 1, where K is the number of clusters, D is the number of data items, *Dim* is the number of attributes.

Table 1. Synthetic Data Sets

Name	K	D	*Dim*	Source
Art1	4	4×250	2	$N([0, 0], [2, 2])$, $N([10, 10], [2, 2])$ $N([0, 10], [2, 2])$, $N([10, 0], [2, 2])$
Art2	4	4×250	2	$N([0, 0], [2, 2])$, $N([6, 6], [2, 2])$ $N([0, 6], [2, 2])$, $N([6, 0], [2, 2])$
Art3	4	769, 77, 77, 77	2	$N([0, 0], [2, 2])$, $N([10, 10], [2, 2])$ $N([0, 10], [2, 2])$, $N([10, 0], [2, 2])$

Art1, Art2, Art3 is same as square1, square5, sizes5 of the data set used in [8].

Table 2 shows the Iris data set of the machine learning repository (http://www.

ics.uci.edu/~mlearn/MLRepository) that is also used in our experiments.

Table 2. Real Data Set

Name	*K*	*D*	*Dim*
Iris	3	3×50	4

We compare the performance of our proposed algorithm with the ant-based algorithm, ATTA in [10]. ATTA is a much improved algorithm in clustering speed and robustness compared to the existing algorithm in [7]. The clustering results are evaluated in clustering error and time. We also compared the algorithm in [5] with our SLAC but omit the comparison results in this paper because it is inferior to ATTA.

Ant-based clustering algorithm is very sensitive to the value of α. In SLAC the search ant and labor ant use different values of α. It is designed to be careful for search ant's dropping data items and promoted for labor ant's quick dropping data items. We set empirically $\alpha_S = 0.32$ for search-ant and $\alpha_L = 0.8$ for labor ant for the synthetic data sets. However, for the Iris data set, $\alpha_S = 0.28$, $\alpha_L = 0.7$, is used respectively. We set initial $\alpha = 0.5$ in ATTA.

For the synthetic data sets or the Iris data set, we already knew the numbers of the correct clusters. We count the number of clusters obtained from ATTA and SLAC for the comparison. Note that a cluster merge operation can be carried out, if needed, based on the similarities between clusters after the termination of ATTA or even for our algorithm SLAC where the number of clusters is initially given. If the numbers of clusters from the algorithms are not consistent to the known numbers, we did not use those cases to calculate the mean and the standard deviation of clustering errors. Computing times in Table 3, 4, 5, and 6 are measured excluding the time for graphical output generation. A computer with Pentium 4 2.8GHz CPU, and 256M memory is used for Visual C++ programming. The experiment was carried out for 30 times for each data set. Table 3 to 6 are representing the results.

4.2 Computational Results

The performance of ant-based clustering varies with the magnitude of overlapping among the data items between the clusters. Art1 data set has four clusters clearly since its data are overlapped barely. From the Table 3, average number of clusters is

consistent to the proper number (i.e., 4) in both ATTA and SLAC in 30 trials. Both algorithms produce also small clustering errors. The number of iterations in ATTA is set to 300,000 by analyzing some pilot trials. The SLAC has similar clustering errors and less computing time even with a fewer number of iterations.

Table 3. Test Result for Art1 Data Set

	ATTA	SLAC
Iteration	300,000	50,000
Average # of cluster	4	4
# of finding correct clusters	30	30
Mean of clustering error(%)	1.4	1.6
Std of clustering error(%)	0.7	0.4
Mean of computing time(sec)	6.79	2.92

Art2 data set is more overlapped than Art1 data set which implies insignificance of clustering errors. In Table 4, we can see ATTA made average of 3.34 clusters even with 1,000,000 iterations. SLAC sometimes made 3 clusters by merging two clusters into one cluster. If we increase the nest memory of SLAC from 20 to 50, the average number of clusters increased also the standard deviation of clustering error and the clustering time. We can know that the more data items assigned by search ants initially, the higher chance for clusters will not be merged by increasing the nest memory size. SLAC has bigger standard deviation of clustering errors than ATTA for this data set. It can be explained as we did not include those cases to calculate the mean and standard deviation values of clustering errors, if the number of clusters from the algorithms is not consistent to the known numbers.

Table 4. Test Result for Art2 Data Set

	ATTA	SLAC (memory = 20)	SLAC(memory = 50)
Iteration	1,000,000	50,000	50,000
Average # of clusters	3.36	3.87	3.93
# of finding correct clusters	21	26	28
Mean of clustering error(%)	17.1	15.7	15.2
Std of clustering error(%)	0.9	3.1	4.8
Mean of computing time(sec)	14.68	2.98	5.16

Art3 data set has four clusters containing data items of 769, 77, 77, and 77. There is one big cluster and three small clusters for the Art3 data set. Table 5 shows weakness of SLAC. For a data item from a bigger data set, it has higher probability than the data items of smaller data set since search ants pick the data items randomly. Those picked data items are considered as the data items in the nest memory. There is a high chance of merging if two storage nests are filled with similar data items in the early stage of forming the nest memory. The ATTA performs well on Art3 data set, but SLAC sometimes failed to keep the number of clusters for this data set.

Table 5. Test Result for Art3 Data Set

	ATTA	SLAC
Iterations	1,000,000	50,000
Average # of clusters	4	3.8
# of finding correct clusters	30	24
Mean of clustering error(%)	1.5	2.1
Std of clustering error(%)	0.5	2.2
Mean of computing time (sec)	14.52	4.41

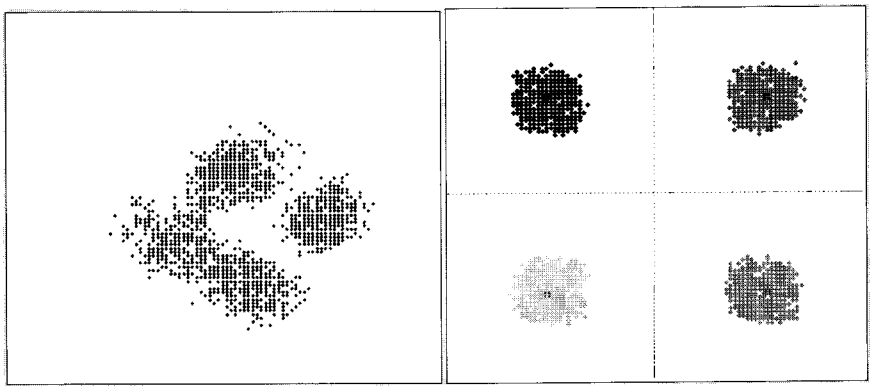
In Table 6, the ATTA never generated three clusters with Iris data set. Since ATTA is very sensitive to values of α, we tried various α values (0.3~0.8), but ATTA failed to find the correct number of clusters every time. The SLAC completed the clustering of Iris data set which has 150 data items in average clustering time of 0.15 seconds with 8% mean of clustering errors.

Table 6. Test Rresult for Iris Data Set

	ATTA	SLAC
Iteration	1,000,000	5,000
Average # of cluster	2	3
# of finding correct clusters	0	30
Mean of Clustering error(%)	Not Available	8
Std of Clustering error(%)	Not Available	1.2
Mean of computing time(sec)	6.25	0.15

Figure 3 (a) is one of the clustering results by ATTA with Art2 data set. It looks

like there are three or four clusters. However there is only one cluster identified by ATTA. The problem of ATTA is originated by the identification method of clusters is based on the distance of a lattice. The four clusters should move away from each others fon the lattice to increase the power of identification of the clusters. However, sometimes the masses are prone to get closer while moving along. Increasing the number of iterations does not guarantee better performance for this problem. In SLAC, this phenomenon does not occur. Since the cluster identification method is different, as we can see in Figure 3 (b), all clusters are formed distant from each other.



(a) Clustering from ATTA (1,000,000 iter.) (b) Clustering from SLAC (50,000 iter.)

Figure 2. Comparison of Clustering Results of SLAC and ATTA

5. Concluding Remark

In this paper, we present investigation on basic ant-based clustering algorithm and propose a search-ant and labor-ant clustering (SLAC) algorithm that is based on mimicking the real ant's behavior. The proposed algorithm used search ants and labor ants to balance their work load and reduce the traveling path of labor ants.

The performance of SLAC algorithm is faster than existing ant-based clustering algorithms and can be applied to various applications. SLAC can also set the number clusters for a special purpose usage. For the application that has no preassigned number of clusters, users can try several trials without hesitation to find an optimal number of clusters by utilizing the speed of SLAC.

Initial data items assigned by search ants of SLAC contribute a significant influ-

ence toward following clustering process, having high standard deviation of clustering errors for certain data set, are remaining problems to be tackled. The rationale for setting α values and the size of nest memory is also needed to improve SLAC.

Reference

[1] Dorigo, M. and G. Di Caro, *The ant colony optimization metaheuristic*, In Corne, D., Dorigo, M., and Glover, F., (Eds.), New Ideas in Optimization, (1999), 11-32, London, UK.

[2] Bonabeau, E., M. Dorigo, and G. Theraulaz, Swarm Intelligence-From Natural to Artificial Systems, Oxford University Press, New York, NY, USA, 1999.

[3] Dorigo, M., E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems* 16, 8 (2000), 851-871.

[4] Deneubourg, J.-L., S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chr'etien, *The dynamics of collective sorting: Robot-like ants and ant-like robots*, In Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 1 (1991), 356-365, Cambridge, MA, MIT Press.

[5] Lumer, E. and B. Faieta, *Diversity and adaptation in populations of clustering ants*, In Proceedings of the Third International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3 (1994), 501-508, Cambridge, MA, MIT Press.

[6] Kuntz, P., P. Layzell, D. Snyder, *A colony of ant-like agents for partitioning in VLSI technology*, in P. Husbands, I. Harvey (Eds.), Proceedings of the Fourth European Conference on Artificial Life, (1997), 412-424, MIT Press, Cambridge, MA.

[7] Handl, J. and B. Meyer, *Improved ant-based clustering and sorting in a document retrieval interface*, PPSN VII, LNCS 2439, 2002.

[8] Handl, J., J. Knowles, and M. Dorigo, *Strategies for the increased robustness of ant-based clustering*, In Engineering Self-Organising Systems, Lecture Notes in Computer Science, 2977 (2004), 90-104, Heidelberg, Germany: Springer-Verlag.

[9] Handl, J., J. Knowles, and M. Dorigo, *Ant-based Clustering and Topographic Mapping*, Artificial Life, 12, 1 (2004).

[10] Handl, J., J. Knowles, and M. Dorigo, *Ant-based Clustering and Topographic Mapping*, Artificial Life, 12, 1 (2004).

[11] N. Labroche, N. Monmarch´e, and G. Venturini, *A new clustering algorithm based on the chemical recognition system of ants*, in Proc. ECAI-02, Lyon FRANCE, (2002), 345-349.

[12] Labroche, N., C. Guinot, and G. Venturini, *Fast Unsupervised Clustering with Artificial Ants*, PPSN VIII, LNCS 3242, (2004), 1143-1152.