

Ant Colony Optimization with Dual Pheromone Tables for Clustering

Chun-Wei Tsai
Department of Applied
Geoinformatics
Chia Nan University of
Pharmacy & Science
Tainan, Taiwan, R.O.C.
Email: cwttsai0807@gmail.com

Kai-Cheng Hu and Ming-Chao Chiang
Department of Computer Science
and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan, R.O.C.
Email: spring69953@hotmail.com
and mcchiang@cse.nsysu.edu.tw

Chu-Sing Yang
Department of Electrical
Engineering
National Cheng Kung
University
Tainan, Taiwan, R.O.C.
Email: csyang@ee.ncku.edu.tw

Abstract—This paper presents a novel pheromone update strategy for improving the clustering results of ant colony optimization (ACO). The proposed algorithm is motivated by the observation that most of the ACOs only keep track of the *promising* foraging information, which has the potential to lead to better solutions than all the other search directions in the pheromone table. This eventually makes the search converge to particular search directions in later iterations because the pheromone values on good routing paths will be reinforced. As such, the breadth of search (diversity) will be reduced, thus limiting the clustering results of ACO. The proposed algorithm adds a second pheromone table to ACO for recording the *unpromising* foraging information that is worse than all the other search directions and using a novel construction method to explore the new search directions. In other words, by leveraging the strengths of diversification and intensification, the proposed algorithm can find better solutions than traditional ACO. To evaluate the performance of the proposed algorithm, we use it to solve the data clustering problem. Our experimental results indicate that the proposed algorithm can significantly improve the quality of ant colony optimization.

Keyword: ant colony optimization; pheromone table, and clustering.

I. INTRODUCTION

Partitional clustering is a traditional clustering approach [1], [2], [3] that partitions n patterns in d -dimensional space into k groups according to some predefined criteria. The ultimate goal of clustering is to find a partitioning that minimizes the distance between all patterns and their centroids while at the same time maximizing the distance between clusters. In practice, the most popular metric used to measure the clustering results is the sum of squared errors (SSE) defined as

$$SSE = \sum_{i=1}^k \sum_{j=1}^{n_i} \|x_{ij} - c_i\|^2, \quad (1)$$

where k denotes the number of clusters; n_i the number of patterns in the i th cluster; x_{ij} the j th pattern in the i th cluster; $c_i = (1/n_i) \sum_{j=1}^{n_i} x_{ij}$ the mean (centroid) of the i th cluster; and $n = \sum_{i=1}^k n_i$ the number of patterns. On the other hand, metrics other than SSE such as f-measure or entropy can be

used when the same clustering algorithm is applied to solve other clustering problems.

Clustering technologies are important because many applications such as search engine [4], [5], [6], document analysis [7], [8], bio-information [9], [10], financial analysis [11], and human face recognition [12], [13] rely on the clustering technologies. More recently, how to find an approximate solution to a clustering problem, which is NP-hard [14], in a reasonable time has become a very important research topic partially because the dataset size has become larger and larger and partially because finding the optimum solution to a clustering problem takes an extraordinary amount of time.

Since 1950's, many methods have been proposed to solve the clustering problem. Some of them are deterministic, some stochastic, and some statistic. The k -means [15], [16] algorithm is undoubtedly one of the most well known clustering algorithms because it is very easy to implement and it will converge to a local optimum very quickly. However, the clustering results of k -means are not satisfied for many applications because the k -means algorithm has several drawbacks. Of them are: the final result is very sensitive to the initial solutions; it is incapable of separating clusters that are non-linearly separable; and it may fall into local minima at early iterations. In addition to k -means, metaheuristics provide another way for clustering the data to mitigate the problems of k -means. Over the past, many metaheuristics have been employed to solve the clustering problem such as genetic algorithm (GA) [17], [18], tabu search (TS) [19], particle swarm optimization (PSO) [20], [21], and ant colony optimization (ACO) [22], [23]. Of them, ACO is among one of the most efficient metaheuristic algorithms because it uses social behavior to search for the clustering solution.

Although ACO can provide better results than other metaheuristic algorithm algorithms, it is still facing the local minima problem on its convergence process because by reinforcing the pheromone values on some routing paths that provide better solutions, the breadth of the search directions is limited. However, solutions on worse routing paths may provide partial solutions of the optimum solution or better search directions

even though the chance is very small. For this reason, ACO-based algorithms may lose partial optimum solutions at the future search process by only reinforcing the promising paths. As such, how to improve the end results has recently become a promising research topic. In this paper, we present a novel ACO-based algorithm based on ant colony system (ACS), called ant colony optimization with dual pheromone tables (ACODPT), that adds a second pheromone table to it for keeping track of a different kind of information for improving the end results. A new recombination method for solution construction is also presented to explore the new search directions that may find better solution of clustering. To evaluate the performance of the proposed algorithm, we compare it to three state-of-the-art clustering algorithms: standard k -means [16], ant colony system [24], and genetic k -means algorithm [17].

The remainder of the paper is organized as follows. Section II gives a brief introduction to the genetic algorithm and approaches taken to enhance its performance. Section III describes in detail the proposed algorithm. Also given in Section III is a simple example to demonstrate how the proposed algorithm works. Performance evaluation of the proposed algorithm is presented in Section IV. Conclusion is drawn in Section V.

II. RELATED WORK

Generally speaking, the clustering algorithms [25] that are inspired by the behavior of ants can be divided into two categories: ant-based clustering algorithm (ACA) [26], [27], [28], [29] and ant colony optimization (ACO) [30], [24], [31], [32]. In this section, we will give a brief introduction to these clustering algorithms.

A. Ant-based clustering algorithms

The ant-based clustering algorithm is inspired by the behavior of ant colonies in clustering their corpses and sorting their larvae [33]. The initial study was done by Deneubourg et al. [26] in which they proposed a model to make the ants move randomly and used the behavior of ants in “picking up” and “dropping” patterns to cluster the patterns. In [27], Lumer and Faieta extended the study in [26] to improve the quality of the clustering results. Generally, the clustering algorithms in this class are called the ant-based clustering algorithm (or ACA for short). Fig. 1 shows the basic concept and the major operators used by ACA [27].

```

1 Initialize()
2 while the termination criterion is not met
3   randomly select an ant to move
4    $s = \text{DropItem}()$ 
5    $s = \text{PickupItem}()$ 
6 end
7 OutputResult()
```

Fig. 1. Outline of the ACA.

As shown in Fig. 1, the very first step ACA takes is to initialize all the parameters, randomly scatter data items on the toroidal grid, and randomly place ants on the toroidal grid.

The *DropItem* operator checks to see if ant j carries a pattern and is placed at a free grid position. It decides probabilistically whether to drop the pattern at the current position by Eqs. (2) and (4). The *PickupItem* operator checks to see if ant j does not carry any pattern and has arrived at a grid position occupied by a data pattern. It decides probabilistically whether to pick up this pattern by Eqs. (3) and (4).

$$p_{\text{drop}}(i) = \begin{cases} 2f(i) & \text{if } f(i) < k^-, \\ 1 & \text{otherwise,} \end{cases} \quad (2)$$

$$p_{\text{pick}}(i) = \frac{k^+}{k^+ + f(i)}, \quad (3)$$

$$f(i) = \max \left\{ 0, \frac{1}{\sigma^2} \sum_{j \in L} \left(1 - \frac{d(i, j)}{\alpha} \right) \right\}. \quad (4)$$

where k^+ and k^- are constants, $f(i)$ is a neighborhood function, $d(i, j)$ is the similarity measurement between patterns i and j , α is a data-dependent scaling parameter, and σ^2 is the size of the local neighborhood L .

In [33], Yang and Kamel presented a parallel model by using the multi-ant colonies model that contains independent ant colonies and a queen ant. The proposed three information exchange methods between the queen ant and ant colonies. They are: keeping predecessor, circular exchange, and lowest outliers. The experimental results show that the multi-ant colonies algorithm [33] outperforms single-ant-based algorithm and k -means in terms of several different criteria. In [34], Boryczka presented an extended version of ACA, by modifying the density function $f(i)$ to solve the problem in which the stable value of neighborhood scaling parameter may sometimes cause inappropriate behavior. Another modification is the new α -adaptation scheme that uses simulated annealing to determine its value. In summary, the advantages of ACA are in that it can automatically determine the number of clusters and the clustering results outperforms the statistical methods [25].

B. ACO-based algorithms for clustering

Ant Colony Optimization (ACO) [30], [24], [31], [32] is an efficient method proposed in the early 1990s for attacking the combinatorial optimization problems. It is inspired by the foraging behavior of real ant colonies. Fig. 2 shows the basic concept and the major operators used by ACO to find good solutions.

```

1 Initialize()
2 while the termination criterion is not met
3    $s = \text{SolutionConstruction}()$ 
4    $\tau = \text{PheromoneUpdate}()$ 
5    $\text{LocalSearch}(s)$  // optional
6 end
7 OutputResult()
```

Fig. 2. Outline of the ACO.

As shown in Fig. 2, the very first step ACO takes is to initialize all the parameters such as the best-so-far solution, the iteration-best solution, and the pheromone values τ . After that, the main procedure on lines 2 through 6 will be repeated until the stop condition is met. In general, the main procedure is composed of two required operators *SolutionConstruction()* and *PheromoneUpdate()* and one optional operator *LocalSearch()*.

The *SolutionConstruction()* operator on line 3 that calculates the probability p_{ij}^k for choosing the next sub-solution is defined as follows:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{j \in \mathcal{N}_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta} & \text{if } j \in \mathcal{N}_i^k, \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

where $\mathcal{N}_i^k$ denotes the set of feasible (or candidate) sub-solutions that can be the next sub-solution of i ; τ_{ij} the pheromone value between the sub-solutions i and j ; η_{ij} a heuristic value, which is also referred to as the *heuristic information*, used to denote the quality of sub-solution s_{ij}^k that will affect each ant's determination to move to j when at i ; and α and β the weight of exploration and exploitation.

The *PheromoneUpdate()* operator on line 4 that is employed in updating the pheromone value τ_{ij} on each edge e_{ij} is defined as follows:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho \sum_{k=1}^m \Delta\tau_{ij}^k, \quad (6)$$

$$\Delta\tau_{ij}^k = \frac{1}{L^k} \quad (7)$$

where m denotes the number of ants; L^k either the length of the tour created by ant k , the length of the iteration-best solution, or the length of the best-so-far solution. Normally, local search, such as problem-specific permutation method, is used as an option for improving the quality of the solution of ACO if ACO cannot find an applicable solution.

Recently, several studies [35], [36], [37], [38], [39], [40], [41], [42] have used ACO to solve the clustering problem because it can provide better result than the other traditional clustering algorithms such as k -means. Šilc et al. [35] used the generalized Lloyd algorithm to create a partition, which is then used as the initial solution for ACO to solve the mesh-partitioning problem. Kuo et al. [36], [37] presented another combination to solve the clustering problem that integrates ACO and k -means, called ant k -means algorithm (AK). This algorithm adds four operators to ACO, that is, *divide*, *agglomerate_obj*, *agglomerate*, and *remove*. Initially, all the patterns are grouped into a root cluster. The divide operator plays the role of dividing the root cluster into several new clusters by the predefined criteria. The *agglomerate_obj* operator is then used to agglomerate the patterns into the suitable clusters while the *agglomerate* operator is used to merge clusters that are similar to each other. The *remove* operator is employed to remove patterns that are dissimilar in the cluster to which they belong. The simulation results show that AK [37] outperforms self organizing maps with k -means and self organizing maps with

genetic algorithm. Tsai et al. [38] presented another modified version of ACO to solve the clustering problem by adding several efficient strategies, called ant colony optimization with different favor (ACODF). One of the efficient strategies is that ACODF does not require ant to visit all the patterns (nodes) to be clustered. Another efficient strategy is that ACODF splits patterns the pheromone value between them is less than a predefined threshold into different groups. Li et al. [39] parallelized the ACO for the codebook generation problem (CGP) by using MPI. The simulation results show that the parallel strategy can improve the quality of ACO for CGP by about 0.1 up to 0.2 dB of peak signal-to-noise ratio (PSNR). Han and Shi [40] presented an efficient ant-based algorithm for fuzzy clustering the image segmentation problem. First, k centers are selected based on the image histogram. Then, the gradients of the k centers are used to determine the neighboring features to reduce the computation time.

Ramos et al. [41] combined hyperboxes and ACO to solve the clustering problem, called hyperbox clustering with ACO (HACO). More precisely, the hyperboxes are placed in the search space to reduce the number of patterns that ACO needs to compute. The hyperboxes are then grouped by using the nearest-neighbor (NN) method. The computation time of HACO can then be reduced so that it is faster than ACO. Korürek and Nizam [42] used ACODF [38] to cluster unknown patterns to train the arrhythmia detection system. The nearest neighborhood classifier is used to test the system. The results show that ACO can be used to improve the analysis results of the medical signals such as electrocardiography (ECG).

III. THE PROPOSED ALGORITHM

In this section, we present an efficient ACO-based algorithm based on two pheromone tables, called ant colony optimization with dual pheromone tables (ACODPT). We will begin with a brief introduction to the proposed algorithm and an example of ACODPT before we move on to the details of the proposed algorithm.

A. The concept

In general, most of the ACO-based algorithms use pheromone to record and propagate the *positive information* between ants in the population. Consequently, when one or more ants find a good path (search direction or solution), the other ants in the population will follow this path by a positive feedback to the pheromone table. That is, the ACO-based algorithms only keep the positive information to the pheromone table. As such, the breadth of the search directions will be reduced as the number of iterations increases. In this paper, we add a second pheromone table to lay down the negative information. As shown in Fig. 3, the proposed algorithm, ACODPT, has two different kinds of pheromone tables: the positive pheromone table and the negative pheromone table. After the transition and mean update operator, the pheromone values of some paths in the positive pheromone table will be strengthened when ants routing on these paths find better results than the other ants. At the same time, the pheromone

values of some paths in the negative pheromone table will be increased when ants routing on these paths find worse results than the other ants.

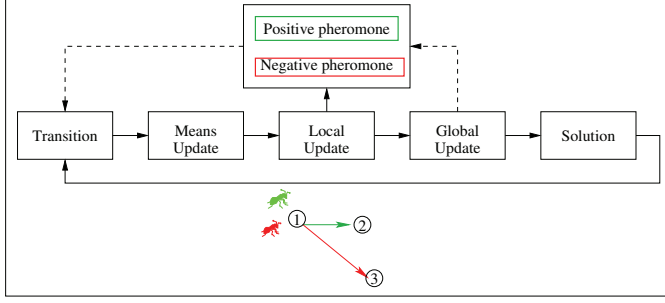


Fig. 3. An illustration of the proposed algorithm.

For example, as depicted at the bottom of Fig. 3, the green ant finds a better route path (starting with node 1 to node 2) than the red ant (node 1 to node 3). The green ant then updates the positive pheromone table to keep its route path whereas the red ant updates the negative pheromone table to keep its route path. Simply speaking, the underlying idea is to prevent the ant from routing the some paths they have already traveled that are worse than paths (solutions) created by the other ants. From this perspective, it is used to avoid only searching the similar directions that can find better result.

B. The ant colony optimization with dual pheromone tables

In order to integrate two pheromone tables into the proposed algorithm, the way the probability is calculated has to be modified, which in turn changes the solution construction operators, as given below.

$$s = \begin{cases} \arg \max_{u \in \mathcal{N}_r^k} \{ [\tau_{ru}] \cdot [\eta_{ru}]^\beta \} & \text{if } q \leq q_0, \\ v & \text{if } q_0 < q \leq q_1, \\ w & \text{otherwise,} \end{cases} \quad (8)$$

$$q_{rz}^k = \begin{cases} \frac{[\tau_{rz}] \cdot [\eta_{rz}]^\beta}{\sum_{u \in \mathcal{J}(r)} [\tau_{ru}] \cdot [\eta_{ru}]^\beta} & \text{if } z \in \mathcal{N}_r^k \setminus B_r^k, \\ 0 & \text{otherwise,} \end{cases} \quad (9)$$

$$B_r^k = B_r^k \cup \{\bar{r}z\} \quad \forall z \in \mathcal{N}_r^k \text{ and } \tau_{rz}^b < q_b, \quad (10)$$

where q is a random value to determine which strategy will be used to construct the next sub-solution, q_0 and q_1 are used to predefine the intervals of the three strategies used in ACODPT. Similar to ACS, as defined in Eq. (8), the first two selection strategies are: select the best next sub-solution (when $q \leq q_0$) and select the next solution with the probability calculated by Eq. (5) (when $q_0 < q \leq q_1$).

When $q > q_1$, the proposed algorithm will, of course, use Eqs. (9) and (10) to determine the next sub-solution. As defined in Eq. (10), B_r^k represents the path that will not be selected, $q_b \in (0, 1)$ is a random value, and $\tau^b \in (0, 1)$ represents the second kind of pheromone on the path. In other words, the path will be included to the set B_r^k if the pheromone value τ^b is less than q_b . As a consequence, the paths (solutions) that

are worse than the other existing paths found by ants will not be selected when the ACODPT uses q_{rz}^k to determine the next sub-solutions.

```

1 Initialization
2 while the termination criterion is not met
3   for each ant  $k$ 
4     1. assign all patterns to the clusters to which they belong by
5       Eqs. (8), (9) and (10).
6     2. update centroids
7     3. local update each path  $\bar{r}s$  ant  $k$  moved
8   end
9   global update each path  $\bar{r}s$  by Eqs. (11) and (12)
10 end
11 Output the best result

```

Fig. 4. Outline of ACODPT

Fig. 4 gives an outline of the proposed algorithm. ACODPT will first initialize the positive pheromone table and the negative pheromone table by setting all the paths to the initial value, followed by setting the parameters, such as α , β , ρ , and SSE to their initial values. Next, ACODPT will enter the main loop, which will be performed until the stop criterion is met. At the solution construction step (lines 3–7), each ant in the population will take care of assigning the input patterns to the clusters by using the equations Eqs. (8), (10) and (9).

In this paper, the proposed algorithm uses the group ID of all patterns to represent the clustering result. As shown in Fig. 5, suppose that we have four input patterns to be partitioned into three groups (clusters). The solution of each ant k of the proposed algorithm can be denoted by $s^k = \{s_1^k, s_2^k, s_3^k, s_4^k\}$, with s_i^k indicating to which cluster pattern i belongs, determined by Eqs. (8), (10) and (9) (see also lines 4 and 5).

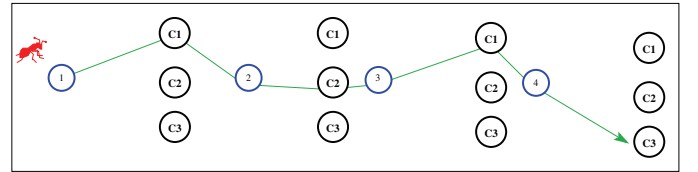


Fig. 5. Example illustrating how ACODPT assigns a pattern to the cluster to which it belongs.

This procedure will be repeated for all the input patterns. The centroids of the solution created by the ant k will then be updated when that ant completed the tour. In addition to local update the pheromone values in the positive pheromone table, ACODPT will also global update the pheromone values in the positive pheromone table defined as

$$\tau_{rs} = (1 - \rho)\tau_{rs} + \rho\Delta\tau_{rs}^{bs}, \quad \forall \bar{r}s \in T^{bs}, \quad (11)$$

where T^{bs} denotes the best-so-far tour and τ^{bs} the pheromone value that will be increased by the global update of this path, and the pheromone values in the negative pheromone table defined as

$$\tau_{rs}^b = \gamma\tau_{rs}^b, \quad \forall \bar{r}s \in T^{iw}, \quad (12)$$

where T^{iw} denotes the iteration-worst tour. Moreover, γ is set to 0.999 as far as this paper is concerned. When the termination criterion is met, ACODPT will output the best-so-far solution as the final solution.

IV. SIMULATION RESULTS

In this section, the performance of the proposed algorithm is evaluated by comparing it with several state-of-the-art clustering algorithms. But before that, we will first give the parameter settings and datasets used.

A. Parameter settings and datasets

To evaluate the performance of ACODPT, we compare it with three state-of-the-art clustering algorithms: standard k -means [16], genetic k -means algorithm [17], and ant colony system [24]. The empirical analysis is conducted on an Intel i7 CPU 920 with 2.67 GHz CPU and 4GB of memory running on Fedora 12 with Linux 2.6.31.5-127. Also, the programs are written in C and compiled by gcc version 4.4.2 20091027 (Red Hat 4.4.2-7). Moreover, for GKA, the population size is fixed at 25, so is the population size of ACS (i.e., the number of ants). Same as the GKA, the crossover operator is replaced by a 2-iteration k -means operator, the mutation probability is 0.01, and the number of generations is 1,000. For the ACS and ACODPT, α is set to 0.1, β to 11.0, ρ to 0.1, and τ to 0.01. q_0 and q_1 are set to 1 and 75, respectively. To avoid premature convergence, the proposed method is started at iteration 2. For comparison, the datasets used in this paper to evaluate the performance of these clustering algorithms are iris, wine, yeast, and abalone, all from [43].

B. Results

Table I compares the performance of ACODPT with the following clustering algorithms: k -means, GKA, and ACS. It shows that GKA provides a better result than k -means while ACS providing a better result than GKA for the benchmarks yeast and abalone. In contrast, our observation shows that the proposed algorithm, ACODPT, provides a better result than ACS because these two benchmarks (yeast and abalone) are more difficult than the others (iris and wine). For this reason, the proposed algorithm can further improve the quality of ACS by about 1.43% and 29.24% for the benchmarks ecoli, yeast, and abalone. Moreover, although the clustering results of ACODPT of benchmarks wine, ecoli, and yeast are worse than GKA, we believe that it is because the ACS and the proposed algorithm do not use the local search to fine tune the clustering results but GKA does.

V. CONCLUSION

In this paper, we proposed an efficient method, called ACODPT, to improve the quality of ACS-based clustering algorithms. The major concept is to add a negative pheromone table to record the route paths which may lead the ants to create worse solutions. It is inspired by how to prevent the ants from falling into the local minima region again and again. For this reason, the solution construction operator is modified to take into account the way the strategies are selected. In addition

TABLE I
SIMULATION RESULTS

Data set	$x \times y \times z$	k -means	GKA
iris	$3 \times 4 \times 150$	85.34	78.94
wine	$3 \times 13 \times 178$	2,370,690.00	2,370,689.69
glass	$9 \times 6 \times 214$	394.58	340.82
ecoli	$8 \times 7 \times 336$	2,284.85	1,815.70
yeast	$10 \times 8 \times 1484$	49.06	45.83
abalone	$29 \times 8 \times 4177$	1,144.10	572.99
Data set	$x \times y \times z$	ACS	ACODPT
iris	$3 \times 4 \times 150$	78.94	78.94
wine	$3 \times 13 \times 178$	2,370,690.00	2,370,690.00
glass	$9 \times 6 \times 214$	336.06	336.06
ecoli	$8 \times 7 \times 336$	2,579.26	1,995.67
yeast	$10 \times 8 \times 1484$	47.44	46.30
abalone	$29 \times 8 \times 4177$	484.93	482.57

x , y , z denote, respectively, the number of clusters, attributes, and patterns.

to a simple example given to show the design concept, we also describe in detail the proposed algorithm. Our simulation results showed that the proposed algorithm outperforms all the other clustering algorithms compared in this paper in terms of the quality of the end result, especially for large scale datasets. Our future goal is to reduce the computation time of the proposed algorithm while at the same time improving the quality of the end result such as local search methods..

ACKNOWLEDGMENT

This work was supported in part by the National Science Council, Taiwan, R.O.C., under Contract Nos. NSC100-2218-E-041-001-MY2 and NSC99-2221-E-110-052.

REFERENCES

- [1] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Englewood Cliffs, NJ: Prentice Hall, 1988.
- [2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, 1999.
- [3] R. Xu and D. W. II, "Survey of clustering algorithms," *IEEE Transaction on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.
- [4] F. Giannotti, M. Nanni, D. Pedreschi, and F. Samaritani, "Webcat: Automatic categorization of web search results," *Proceedings of SEBD'2003*, pp. 507–518, 2003.
- [5] P. Ferragina and A. Gulli, "A personalized search engine based on web-snippet hierarchical clustering," *International World Wide Web Conference (WWW2005)*, pp. 801–810, 2005.
- [6] H. J. Zeng, Q. C. He, Z. Chen, W. Y. Ma, and J. Ma, "Learning to cluster web search results," *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'04)*, pp. 210–217, 2004.
- [7] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267–273, 2003.
- [8] A. Leuski, "Evaluating document clustering for interactive information retrieval," in *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pp. 33–40, 2001.
- [9] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler, "Knowledge-based analysis of microarray gene expression data by using support vector machines," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97, no. 1, pp. 262–267, 2000.
- [10] G. Getz, H. Gal, I. Kela, D. A. Notterman, and E. Domany, "Coupled two-way clustering analysis of breast cancer and colon cancer gene expression data," *Bioinformatics*, vol. 19, pp. 12079–12084, 2003.

- [11] H. Lawrence, "Face recognition: A literature survey," *Review of Financial Studies*, vol. 4, no. 3, pp. 389–415, 1991.
- [12] R. W. Zhao, Chellappa, P. J. Philips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.
- [13] T. Fromherz, P. Stucki, and M. Bichsel, "A survey of face recognition," *MML Technical Report*, 1997.
- [14] P. Brucker, "On the complexity of clustering problems," *Lecture Notes in Economics and Mathematical Systems*, vol. 157, pp. 45–54, 1978.
- [15] A. L. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.
- [16] J. B. McQueen, "Some methods of classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.
- [17] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Transactions on System, Man and Cybernetics—Part B: Cybernetics*, vol. 29, no. 3, pp. 433–439, 1999.
- [18] S. Bandyopadhyay and U. Maulik, "An evolutionary technique based on k-means algorithm for optimal clustering in R^N ," *Information Sciences*, vol. 146, no. 1–4, pp. 221–237, 2002.
- [19] M. K. Ng and J. C. Wong, "Clustering categorical data sets using tabu search techniques," *Pattern Recognition*, vol. 35, no. 12, pp. 2783–2790, 2002.
- [20] D. van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *Proceedings of the 2003 Congress on Evolutionary Computation*, pp. 215–220, 2003.
- [21] X. Cui, T. E. Potok, and P. Palathingal, "Document clustering using particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 185–191, 2005.
- [22] R. J. Kuo, L. M. Ho, and C. M. Hu, "Integration of self-organizing feature map and k-means algorithm for market segmentation," *International Journal of Computers and Operations Research*, vol. 29, no. 11, pp. 1475–1493, 2002.
- [23] A. Ghosh, A. Halder, M. Kothari, and S. Ghosh, "Aggregation pheromone density based data clustering," *Information Sciences*, vol. 178, no. 13, pp. 2816–2831, 2008.
- [24] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
- [25] J. Handl and B. Meyer, "Ant-based and swarm-based clustering," *Swarm Intelligence*, vol. 1, no. 2, pp. 95–113, 2007.
- [26] J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien, "The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot," in *Proceedings of the First Conference on Simulation of Adaptive Behavior: From Animals to Animats* (J. Meyer and S. Wilson, eds.), (Cambridge, MA), pp. 356–365, MIT Press, 1991.
- [27] E. D. Lumer and B. Faieta, "Diversity and adaptation in populations of clustering ants," in *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3: from animals to animats 3*, pp. 501–508, MIT Press, 1994.
- [28] J. Handl and B. Meyer, "Improved ant-based clustering and sorting," in *Proceeding of 7th International Conference Parallel Problem Solving from Nature - PPSN VII*, pp. 913–923, 2002.
- [29] J. Handl, J. Knowles, and M. Dorigo, "Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1d-som," in *Proceedings of the Third International Conference on Hybrid Intelligent Systems*, IOS Press, 2003.
- [30] M. Dorigo, V. Maniezzo, and A. Coloni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, vol. 26, no. 1, pp. 29–41, 1996.
- [31] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.
- [32] M. Dorigo and T. Stützle, *Ant Colony Optimization*. The MIT Press, 2004.
- [33] Y. Yang and M. S. Kamel, "An aggregated clustering approach using multi-ant colonies algorithms," *Pattern Recognition*, vol. 39, no. 7, pp. 1278–1289, 2006.
- [34] U. Boryczka, "Finding groups in data: Cluster analysis with ants," *Applied Soft Computing*, vol. 9, no. 1, pp. 61–70, 2009.
- [35] J. Šilc, P. Korosčec, and B. Robič, "Combining vector quantization and ant-colony algorithm for mesh-partitioning," in *Proceedings of 5th International Conference Parallel Processing and Applied Mathematics (PPAM)*, pp. 113–118, 2003.
- [36] R. J. Kuo, C. L. Cha, S. H. Chou, C. W. Shih, and C. Y. Chiu, "Integration of ant algorithm and case based reasoning for knowledge management," in *Proceedings of International Conference on IIJE*, 2003.
- [37] R. J. Kuo, H. S. Wang, T.-L. Hu, and S. H. Chou, "Application of ant k-means on clustering analysis," *Computers & Mathematics with Applications*, vol. 50, pp. 1709–1724, 2005.
- [38] C.-F. Tsai, C.-W. Tsai, H.-C. Wu, and T. Yang, "Acodf: a novel data clustering approach for data mining in large databases," *Journal of Systems and Software*, vol. 73, pp. 133–145, 2004.
- [39] X. Li, X. Yu, and X. Luo, "Parallel implementation of ant colony optimization for vector quantization codebook design," in *Proceedings of the International Conference on Natural Computation*, vol. 4, pp. 787–791, 2007.
- [40] Y. Han and P. Shi, "An improved ant colony algorithm for fuzzy clustering in image segmentation," *Neurocomputing*, vol. 70, no. 4–6, pp. 665–671, 2007.
- [41] G. N. Ramos, Y. Hatakeyama, F. Dong, and K. Hirota, "Hyperbox clustering with ant colony optimization (haco) method and its application to medical risk profile recognition," *Applied Soft Computing*, vol. 9, no. 2, pp. 632–640, 2009.
- [42] M. Korićek and A. Nizam, "Clustering mit-bih arrhythmias with ant colony optimization using time domain and pca compressed wavelet coefficients," *Digital Signal Processing*, vol. 20, no. 4, pp. 1050–1060, 2010.
- [43] UCI Datasets. URL: <http://archive.ics.uci.edu/ml/datasets/Iris>.