A Bio-Inspired Algorithm for Searching Relationships in Social Networks

Jessica Rivero, Dolores Cuadra, Francisco Javier Calle, Pedro Isasi
Computer Science Department
Carlos III University of Madrid
Madrid, Spain
{jrivero, dcuadra, fcalle}@inf.uc3m.es, Isasi@ia.uc3m.es

Abstract: Nowadays the Social Networks are experiencing a growing importance. The reason of this is that they enable the information exchange among people, meeting people in the same field of work or establishing collaborations with other research groups. In order to manage social networks and to find people inside them, they are usually represented as graphs with persons as nodes and relationships between them as edges. Once this is done, establishing contact with anyone involves searching the chain of people to reach him/her, that is, the search of the path inside the graph which joins two nodes. In this paper, a new algorithm based on nature is proposed to realize this search: SoS-ACO (Sense of Smell - Ant Colony Optimization). This algorithm improves the classical ACO algorithm when it is applied in huge graphs.

Keywords: Large graphs, Social Networks, ACO, Dijkstra, Path search

I. Introduction

One of the most important applications that exist nowadays is the Social Networks. This is due to the fact that they are used by the great majority of the people around the globe to contact with other people and to interchange information with them.

These applications could be very useful for researchers to establish collaborations between different groups, or to contact with people interested in the same knowledge area.

To realize these connections it is very important to follow a chain of people with any kind of relation between them to ensure that the friend request is not rejected by the person of interest.

In order to deal with this type of information and to search the way to reach a person, from another one, inside a social network, an efficient way to represent the information is as a graph such that persons are nodes and the relationships between them are links in the graph. Examples of this transformation and its problems can be found in [1] where nodes represent students in Club Nexus (a social network of the Stanford University) and links represent the relation between students (to do that, different factors must be considered such as gender or class year of the student).

Once this is done, to search a person involves to search the path between two nodes of the graph, that is, to apply a search path algorithm in it.

The main problem of this search is that, despite the high amount of algorithm to realize this function, there is not anyone able to handle the size of graphs obtained from social networks using its topology.

Examples of this can be found in [22] which proposal shows a search path algorithm which works efficiently on small graphs (graphs with thousands of nodes). The problem of it is that success rate diminishes when the number of nodes increases. This working way can be found also in [15] which searches paths on graphs on the order of 213 nodes.

To fill the gap in the literature which represents the search of paths in social networks, in this paper a new algorithm is going to be presented.

The base of the proposed algorithm is the Ant Colony Optimization algorithm (ACO [11, 12, 13]). The selection of this algorithm is due to the structure presented by the graphs resulting of the transformation of the Social Networks: graphs with a high degree of clustering and with a small number of steps between any nodes.

Although ACO is normally applied to graphs with hundreds or thousands of nodes, under the conditions previously mentioned it can work fine if some modifications are introduced to deal with the number of nodes.

These modifications are based on the animal behavioral. They use their sense of smell to localize food source and to avoid searching in incorrect areas of the savanna, for example. This sense is going to be applied to ants to avoid invalid search reducing the number of steps that they have to do to reach the destination avoiding that ants get lost and reducing the time to give a path.

To explain the new algorithm, the paper is organized as follows: In Section 2 is going to be shown some relevant works in the area of path searches. In Section 3 the proposal is explained. To test the proposed algorithm, in Section 4 is going to be shown some experiments on real social networks. And finally, Section 5 offers conclusions and future works.

II. Related Works

The search of paths in graphs is not new, but the problem is that the size of the graphs is each time bigger and bigger and the answer to the request of paths must be fast.

Out of the domain of social networks there are a lot of works trying to search paths in large graphs consuming a small period of time.

To do that, all they realize some type of preprocessing to change the structure of the graph and to reduce the number of nodes/links over which the search has to be done. After

that, they apply classical algorithm (A-Star, Dijkstra, etc.) with some modifications to handle the new structure of the graph (hierarchy of subsets of the main graph, a fragmented graph, a shortest path tree, clusters, etc).

Different examples are explained in the following paragraphs attending to the type of storage: main memory and secondary memory.

With respect to the first type of storage mentioned before, [4, 9] organize the graph using a hierarchy, and [8] creates sub-graphs dividing the main graph and taking into account some edges of it which satisfy a fixed characteristic. Usually, this preprocessing takes a lot of time because the number of nodes has to be highly reduced to can be storage in main memory.

In the latter type of storage can be found [5, 6], where a file system is used. In this case, in the preprocessing phase, the graph is divided in fragments and the paths between the nodes in the frontier of those fragments are calculated and stored. In [23, 24], graphs are organized into tree structures and saved in databases. The storage in a database, when a secondary memory is used, is the most typical practice because databases are prepared to handle massive amount of data (More examples of these algorithms can be found in the survey [26]). By that reason, the preprocessing in this case can take a smaller period of time than before.

As all these methods require a large amount of time to complete the pre-processing, it is very difficult to include changes in it as result of the study of the realized queries and their results. That is, these proposals are not adaptable to modifications to the pre-processing factors.

Taking into account the topology of the graphs which represent social networks, it satisfies a very interesting characteristic: the number of steps to reach the destination is low, and there is a high clustering degree. That is, the graph follows a small-world topology [19, 25].

Considering the topology and the adaptability characteristic, there is an algorithm that could be highly recommended: the Ant Colony Optimization Algorithm (ACO algorithm [11, 12, 13]). It has been used in a lot of applications (ACO to solve the problem of job scheduling in grid computing [7], to create personalized guides within museums using mobile devices [16], to determine medical diagnoses [20], etc.), but normally with small graphs (in [2] they use huge graphs but they have the limitations of main memory storage).

In this paper is going to be shown some different experiments of a new version of ACO algorithm previously presented in [21]. The new version is based on a modified ACO in which ants have the ability of detecting the odor of the food. With this modification, that is going to be explained in detail in the next section, the number of links that ants have to use to reach the destination decreases.

The only problem that can be found in ACO is that it does not give the shortest path, but this is not a problem in the domain of social network, the only restriction is to find a path with quality within a certain range (it cannot be very different of the optimum one because as longer the path is, lower is the probability to reach the person destination, because the probability of a negative answer to the friend request of some person in the chain increases).

III. PROPOSAL

Once discussed the different solutions to search paths in huge graphs, in this section it is going to be presented the proposed algorithm to search paths between nodes in social networks, that is, to search relationships between persons.

This version of ACO has the ability of finding the destination node using helps in the graphs. These helps are nodes in the graphs which are commonly used to reach other nodes because they represent persons with a high centrality degree, that is, persons with a lot of friends. This means that there is a high probability of success to reach the destination if that path goes through those nodes.

To detect those nodes, the algorithm is going to deposit in them "food", and it is going to give to each ant the sense of smell to detect the odor of those foods.

With this fact, and because the diffusion of the odor of the food is going to be simulated, ants are able to find these important persons in the graph easily, because it is not necessary to find the food, to find the trail of food odor is enough to know how to reach the food.

As there is a high probability that a lot of ants reach the foods, ones can help others to find the destination node using the trails of pheromones typical of ants.

That is, the "Food nodes" act as a meeting point for ants to make easy to complete the path from the start node to the end node.

An example of the way in which the food is used is shown in Figure 1. In that case, there is only one food. The odor is diffused and reaches to some people of the graph. When a path is requested between the "Start Node" and the "End Node", ants search their destination or a food odor. In the case of Figure 1, they find the odor and use it to reach the destination.

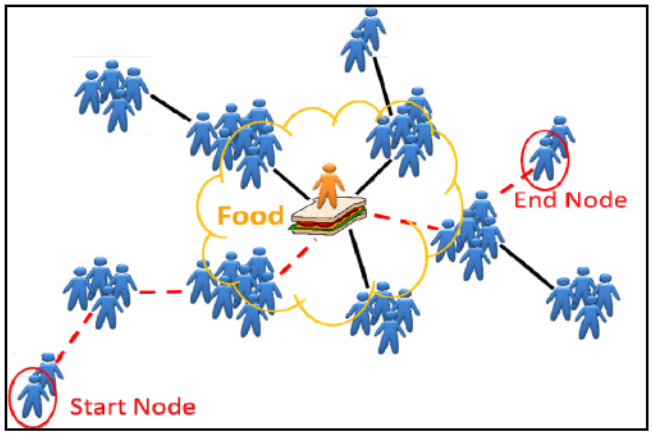

Figure 1. Example of "food" use.

After this explanation, a more formal one is going to be shown in the following subsections (to see the proposed algorithm in more detail [21] can be consulted).

A. Formalization of the problem

Given a particular social network, it is represented by a connected graph that it is defined as $G(N, L, W)$: $N = \{n_i\}$ is a set of nodes; $L \subseteq N \times N$ is a set of edges; and W is a function which gives a positive value for each edge, such that for every edge $l_{ij} = (n_i, n_j) \in L$, there is a value $w_{ij} \in W$.

Over G is going to be carried out some request of paths. These paths between any two nodes of G must be solved in a time less than a value fixed by the user ($t_{threshold}$).

B. Selection of food nodes

The first step in the proposed algorithm is to select the nodes of the graph in which the "Food" is going to be deposited. That is, the selection of the nodes which are going to help ants to reach the destination.

To do this, it is going to be selected the centroids of the graph, that is, the nodes with a high degree of centrality (e.g., celebrities or persons with a lot of friends in the graph).

The resulting set of special nodes is going to be inside of a subset called $F=\{f_i\} \subseteq N$. Each f_i is a food node, and by this reason it has an amount of odor, specifically the biggest amount of odor in the graph is in each f_i. This amount of odor is going to be called m.

As can be observed, this characteristic is only of the nodes, not of the edges.

C. Diffusion of food odor

In order to represent that the odor of a food is not only in the food and that it is diffused by the air in a limited area (the odor decreases the farther the place is from the source), the next step in SoS-ACO is to generate this diffusion in the graph.

To do that, an iterative process is going to be followed, such that the odor of each node is equal to the odor of the previous node minus a factor proportional to the cost of the link which joins them. The start node of the diffusion is each f_i.

The formula followed to do that is $O(n_i)=O(n_j)-k \cdot w_{ij}$, where n_i is the node to which the food odor is diffused from node n_j, $O(n_i)$ and $O(n_j)$ represent the odor in n_i and n_j, w_{ij} represents the cost of the edge which joins both nodes, and the factor k determines the weight of each edge's cost in the decrease.

The odor is set to all nodes which obtain an odor bigger or equal than a given threshold u, and when there are not more nodes with possibility to satisfy this condition, this second step finishes.

After this process, each f_i has an area of odor around it which is called s_i, where each $s_i \subseteq N$ and $S = \{s_i\}$.

If after the diffusion, some studies show that a food node is not necessary, or that other node has to be included in F, the only task that has to be done is to delete the corresponding s_i in the first case, or to create a new one in the second case.

As u determine the size of each s_i, it is very important to select a good value for it, because its value determines the time to create the areas, and by this reason the adaptability of the algorithm.

D. Path searches: modified ACO

The last phase of the proposed algorithm (SoS-ACO) is the search of paths in the graph. To do this, SoS-ACO is based in the ACO algorithm.

The modifications introduced to classic ACO are those which avoid that ants get lost (because the high number of nodes in social networks and, by this reason, the high amount of possibilities in each step) and that make possible the reduction of the answer time. This is done incorporating the different areas of odor around each food node.

Before the search process is explained, it is important to clarify the differences between pheromone and food odor. The first difference is that pheromone is deposited in edges and food odor in nodes. The second, and most important, is that pheromone comes from ants and food odor comes from food nodes (is a characteristic of the graph) although uses ants to diffuse the odor beyond the limits imposed by the value of u.

Once this is clear, the algorithm has, for each request of path, two phases which are going to be described now:

Initialization phase:

- The pheromone of each l_{ij} is reset to a fix value.
- Each ant has a tabu list to avoid that it repeats nodes in its path. In this phase, the tabu list is emptied.
- Ants are divided in two groups: The first group has the nest in the start node of the path and the destination in the end node of the path; and the other has the nest in the end node and the destination in the start node of the path. If one of these end nodes of the path pertains to F, all ants are situated on the opposing non-F node to search for the corresponding f_i.

Path search phase:

Each ant starts its search of the destination node while the execution time does not exceed $t_{threshold}$, and while it does not find its end node.

In each movement of each ant, its tabu list is used to guarantee that the node selected has not been visited before. The only exception to this is the case in which all nodes reachable from the actual node have already been visited. In this case the ant selects one between all the reachable nodes.

The selection between the set of valid nodes (after applied the tabu list condition) for the next step is done calculating the probability shown in Formula 1, where: n_i is the actual node, *NodesR* are the visitable nodes after the application of the tabu list rule, n_j are all nodes with index in *NodesR*, and τ_{ij} is the amount of pheromone in the edge l_{ij}.

$$p(n_i, n_j) = \tau_{ij} / (\Sigma_{k=NodesR} (\tau_{ik})) \quad (1)$$

When a food odor area (s_i) is found, the discoverer ant follows the increasing trail of odor until reach the food node of the area. When this is done, the ant tests if another ant, which follows the opposite direction, found that food before. In the affirmative case, ant joins the part of path found by it (the path from its start node to the food node) and the part of path found by the other ant (path from the food node to its end node).

A verification has to be done each time that a path between an end node of the path and a food node is found with the objective of determining if it has to be stored. This storage is realized if its length is shorter than the previous stored path with the same characteristics of food node and end node or if there is not any path stored with those characteristics.

After this process, the ant continues with its search from the node with food odor found (not from the food node).

With respect to the amount of pheromone deposited in each edge, it is updated each time that an ant founds a complete path (a path between the start node of the path and the end node of the path). This update follows the Formula 2, where $\tau_{ij}(t)$ is the pheromone for the edge l_{ij} in time t, ρ is the dissipation rate of the pheromone (a value between 0 and 1) and *length* is the length of the found path.

$$\begin{aligned} \tau_{ij}(t) &= \tau_{ij}(t-1)\cdot(1-\rho) + \text{constant/length}, \ n_j \text{ in path} \\ \tau_{ij}(t) &= \tau_{ij}(t-1)\cdot(1-\rho), \ n_j \text{ not in path} \end{aligned} \quad (2)$$

After the search finishes, a new diffusion of odor is done if there are paths stored for the food nodes. The diffusion starts by the first node with odor in the path and follows the opposite direction of the path, that is, beginning from the node with food odor and arriving at the node from which the ant originally departed. The food odor decreases according to the same formula described sub-section *C*, that is, $O(n_i)=O(n_j)-k\cdot w_{ij}$.

The only difference between this diffusion of odor and the one described in sub-section *C* is that, in this particular case, the food odor can be smaller than u. Rather, it is necessary that its value simply be greater than zero.

It is important to clarify that if during any of the previous phases any change wants to be introduced in F or in S, it can be realized without affecting the search of paths, because the graph is not restructured, odor is only a help to ants which is applied over the graph without change it.

IV. Evaluation of the Algorithm

After the explanation of the proposed algorithm (SoS-ACO), in this section is going to be realized some experiments to show its proper functioning.

In order to do that some path searches are going to be realized on a real social network graph, and its results in answer time and cost of the obtained path are going to be compared with the results of other algorithms applied over the same graph and with the same searches.

Other thing that wants to be studied is the time required to diffuse the food odor and how the different values of u (threshold for food odor intensity) affect to the previously mentioned parameters.

The selection of the algorithms to compare with is based in the fact that there is not in the state of the art an algorithm centered in the goals of this work, and by this reason it was decided to select well-known and disseminated algorithms:

- *Classic ACO algorithm* [13]. As the base of SoS-ACO is the classical ACO algorithm, could be a good idea to compare the results of the base with the proposal to show the improvements.
- *Dijkstra's algorithm*. As it has been mentioned before, the proposed ACO does not obtain the optimum path. By this reason, this algorithm can be used to determine what is the difference between the quality of the obtained paths with the proposed algorithm and the optimum path (the obtained with Dijkstra). The other reason is that this algorithm is implemented in all the Data Base Management System (DBMS) as the algorithm to search paths in graphs.

With respect to the graph over which the requests of paths are going to be realized, it is a graph which represents the social network Slashdot in February of 2009 [18]. The selection of a social network is because this algorithm has been designed to work in huge graphs with small-world topology (low number of edges between any two nodes of the graph and a high clustering degree).

The number of nodes of this graph is equal to 82,168 and the number of edges is equal to 948,464.

The final thing to completely describe the testing scenario is to explain the type of search paths requests and the amount of them.

A total number of 10 queries with 1,000 services each one are going to be executed sequentially. By service it is going to be understood a request for a path search between two different nodes of the graph selected randomly among all those appearing in the graph. In these experiments only the first solution to each request is going to be selected, and in the moment in which the solution is obtained the search path process finishes.

Once the scenario has been described, it is important to fix some parameters to ACO and SoS-ACO. These are shown in Table I and the explication of each value is the following: $t_{threshold}$ is fixed to the time required by Dijkstra (more or less) to give an answer (in this way the ants never take more time than the time required to obtain the optimum solution), number of ants is equal to 500 because this number gives the best results in the test of scalability showed in [21], pheromone evaporation rate is selected to not have the problem of stagnation in local minimums, m to permit that all the nodes in the graph can have food odor, and k to decrease the amount of odor with a value equal to the cost of the edges.

TABLE I. Algorithm Parameters.

Parameter	Value
$t_{threshold}$	600 sec
#ants	500
ρ	0.6
m	1,000,000
k	100%

With respect to the number of foods which is used in SoS-ACO in this experiment, it is going to be equal to one, that is, there is only one food node ($F = \{f_1\}$), the node representing the person with the greatest number of edges

within the graph, and one food odor area ($S = \{s_1\}$) in the graph.

To finish the description of those things that have effect in the way in which the algorithm works, it is important to say that the food odor in the graph is restarted for every new query, and that the pheromone trails in the graph are restarted at the beginning of every service.

Once the scenario and parameters have been fully described, the results of the experiments are going to be studied.

The first phase is to study the influence of the different u values in the time required to diffuse the food odor. To do this, two values have been selected: $u = 999{,}999$ and $u = 999{,}998$. After the selection of the values of the food odor threshold, the diffusion explained in the sub-section *C* of the previous section is executed. The results of each diffusion can be seen in Table II. They show that with the smaller value of u, the food odor reaches the 37% of the nodes in the graph, and with the bigger one, the percentage of reached nodes decreases to 3%. With respect to the time, it is bigger for the case of 37% of nodes.

TABLE II. AMOUNT OF FOOD ODOR DIFFUSED THROUGHOUT THE GRAPH.

u	Nodes with food odor (%)	Time (sec)
999,999	3	17.4
999,998	37	234.1

The main conclusion extracted from these results is that the kind of preprocessing which uses SoS-ACO in any case is bigger than the time required for the algorithms in the state of the art (which in some cases is equal to days).

With respect to the performance comparison of the proposed algorithm (with 3% initial diffusion and with 37% initial diffusion) with Dijkstra and classic ACO, it is represented in Table III and IV.

Table III shows the information of the cost of the obtained paths for each algorithm. It can be observed that the proposed algorithm's cost is very similar to the optimum obtained by Dijkstra, and that the existent difference is smaller for the proposed algorithm with smaller value of u.

TABLE III. COSTS OF FIRST PATH OBTAINED FOR EACH SERVICE.

	Dijkstra	ACO	Proposed ACO 3%	Proposed ACO 37%
Mean	4,51	387,69	9,43	6,40
Standard Deviation	0,25	76,76	0,86	0,21
Median	4,50	385,74	9,35	6,40

Another conclusion extracted from Table III is that SoS-ACO improves the performance of classic ACO. The difference in cost between them is large despite the fact that the topology of the graph could be considered adequate for the classic ACO (a low number of edges between any two nodes of the graph).

With respect to the time required to give the first solution, the results are shown in Table IV.

It makes clear that the proposed algorithm is superior to the others. With respect to the two versions of the proposed ACO, time is better in the case of 37% than in the other, but the difference is not significant.

Concluding this section, after studying the different experimental results, can be said that SoS-ACO presents a good performance, and that clearly improves the classic ACO algorithm.

TABLE IV. RESPONSE TIMES (SEC) OF FIRST PATH OBTAINED FOR EACH SERVICE.

	Dijkstra	ACO	Proposed ACO 3%	Proposed ACO 37%
Mean	563,39	254,21	1,33	0,59
Standard Deviation	9,69	53,56	0,49	0,14
Median	562,33	254,51	1,22	0,57

V. CONCLUSIONS AND FUTURE WORKS

In this paper, a new version of ACO algorithm, SoS-ACO, has been proposed which is able to search paths in the graphs of Social Networks: graphs with a huge number of nodes and edges, with a high clustering degree, and with a low number of steps to reach the destination node from a start node.

To do this, two new concepts are introduced: the "Food" (characteristic associated to the nodes of the graph with high centrality) and "Food Odor" (used to create areas of odor around the nodes with food and to make finding them easier).

With these two concepts, ants of the ACO algorithm have help to find a path between the start node and the destination node because they use food nodes as meeting points of ants which search the start node of the path and those searching the end node of the path. Such that, after finding the meeting point, the pheromone trail, typical of ACO algorithms, can be used to complete the global path (global path = path from start node to food node + path from food node to end node).

Also, the diffusion of the food odor decreases the number of nodes that has to be explored to find the food node.

To test the way in which this algorithm works, some experiments have been realized over a real social network graph. The results of these experiments are that the proposed algorithm gives paths with a quality very similar to the optimum one, and using a small period of time. That is, SoS-ACO is a good algorithm for the search of relationships between members of a social network.

With respect to the future works, it would be particularly interesting to see how the cost changes when instead of taking the first path, it is taken after a period of time during which ants are running. Additionally, more food nodes can be included and the relation between the number of them and the quality of the solution (in time and cost of the path) can be studied.

Finally, there is an important conclusion which is result of the low time required to diffuse the odor, the fact that odor is only a help and the own characteristics of ACO: SoS-ACO could be used taking into account the dynamic characteristics which are typical in social networks (users

(nodes) and relationships (links) between old and/or new users that continuously appear/disappear).

Due to this reason, in future works, experiments over dynamic graphs want to be realized to test if the new version of ACO can apply the well known good performance of ACO in dynamic environments (examples could be seen in ad-hoc networks [10], in travelling salesman problem [3], in electrical distribution systems management problem [14] or in dynamic vehicle routing [17]) to huge graphs.

ACKNOWLEDGMENT

This study was funded through a competitive grant awarded by the Spanish Ministry of Education and Science for the THUBAN Project (TIN2008-02711) and through MA2VICMR consortium (S2009/TIC-1542, http://www.mavir.net), a network of excellence funded by the Madrid Regional Government.

REFERENCES

[1] L. Adamic and E. Adar, "How to search a social network", Social Networks vol. 27 (3), pp. 187-203, 2005.

[2] E. Alba and F. Chicano, "ACOhg: Dealing with huge graph". Proc. Genetic and Evolutionary Computation Conference of 2007, pp. 10-17, 2007

[3] D. Angus and T. Hendtlass, "Dynamic Ant Colony Optimisation", Applied Intelligence vol. 23 (1), pp. 33-38, 2005.

[4] H. Bast, S. Funke, D. Matijevic, P. Sanders and D. Schultes, "In transit to constant shortest-path queries in road networks", Proc. Workshop on Algorithm Engineering and Experiments of 2007, 2007.

[5] E.P.F. Chan and H. Lim, "Optimization and evaluation of shortest path queries", VLDB Journal vol. 16 (3), pp. 343-369, 2007.

[6] E.P.F. Chan and J. Zhang, "A fast unified optimal route query evaluation algorithm", Proc. 16th ACM conference on Conference on information and knowledge management, pp. 371-380, 2007.

[7] R-S Chang, J-S Chang and P-S Lin, "An ant algorithm for balanced job scheduling in grids". Future Generation Computer Systems vol. 25 (1), pp. 20-27, 2009.

[8] D. Delling, M. Holzer, K. Müller, F. Schulz F, D. Wagner, "High-performance multi-level routing", Series in Discrete Mathematics and Theoretical Computer Science vol. 74, pp. 73-92, 2009.

[9] D. Delling, P. Sanders, D. Schultes and D. Wagner, "Highway hierarchies star", The Shortest Path Problem: Ninth DIMACS Implementation Challenge, DIMACS Book vol. 74, pp. 141-174, 2006.

[10] G. Di Caro, F. Ducatelle and L.M. Gambardella, "AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks", European Transactions on Telecommunications, Special Issue on Self Organization in Mobile Networking vol. 16 (5), pp. 443-455, 2005.

[11] M. Dorigo, "Optimization, learning and natural algorithms", Doctoral Thesis, Dipartamento di Elettronica, Politecnico di Milano, Italy, 1992.

[12] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey", TCS: Theoretical Computer Science vol. 344, pp. 243-278, 2005.

[13] M. Dorigo and T. Stützle, "Ant colony optimization", The MIT Press, 2004.

[14] S. Favuzza, G. Graditi and E. Sanseverino, "Adaptive and Dynamic Ant Colony Search Algorithm for Optimal Distribution Systems Reinforcement Strategy", Applied Intelligence vol. 24 (1), pp. 31-42, 2006.

[15] Guofu Feng, Chunhong Li, Qing Gu, Sanglu Lu and Daoxu Chen, "SWS: Small World Based Search in Structured Peer-to-Peer Systems", Proc. International Conference on Grid and Cooperative Computing Workshops of 2006, pp. 341-348, 2006.

[16] J. Jaén, J.A. Mocholí, A. Catalá and E. Navarro, "Digital ants as the best cicerones for museum visitors", Applied Soft Computing vol. 11 (1), pp. 111-119, 2011.

[17] C-Y Lee, Z-J Lee, S-W Lin and K-C Ying, "An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem", Applied Intelligence vol. 32 (1), pp. 88-95, 2010.

[18] J. Leskovec, "SNAP: Network datasets: Slashdot social network. Stanford University", http://snap.stanford.edu/data/soc-Slashdot0902.html. Accessed 09 November 2010.

[19] M.E.J. Newman, "The structure and function of complex networks", SIAM Review vol. 45 (2), pp. 167-256, 2003.

[20] 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 (2), pp. 632-640, 2009.

[21] J. Rivero, D. Cuadra, J. Calle and P. Isasi, "Using the ACO algorithm for path searches in social networks", Applied Intelligence, DOI: 10.1007/s10489-011-0304-1, 2011.

[22] O. Sandberg, "Distributed routing in small-world networks", Proc. 8th Workshop on Algorithm Engineering and Experiments, pp. 144-155, 2006.

[23] J. Sankaranarayanan and H. Samet, "Distance oracles for spatial networks", Proc. 25th IEEE International Conference on Data Engineering, pp. 652-663, 2009.

[24] J. Sankaranarayanan, H. Samet and H. Alborzi, "Path oracles for spatial networks", Proc. 35th International Conference on Very Large Data Bases, pp. 1210-1221, 2009.

[25] L. Tang and H. Liu, "Graph Mining Applications to Social Network Analysis", In: C. Aggarwal and H. Wang, "Managing and Mining Graph Data", Advances in DataBase Systems vol. 40, pp. 487-514, 2010.

[26] J. Xu Yu, J. Cheng, "Graph Reachability Queries: A Survey", In: C. Aggarwal and H. Wang. "Managing and Mining Graph Data", Advances in DataBase Systems vol. 40, pp. 181-215, 2010.