International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 2, April 2018, pp. 1122~1130
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i2.pp1122-1130

Bat-Cluster: A Bat Algorithm-based Automated Graph Clustering Approach

Zakaria Boulouard[1], Amine El Haddadi[2], Fadwa Bouhafer[3], Anass El Haddadi[4], Lahcen Koutti[5], Bernard Dousset[6]
[1,5]LabSIV, Faculty of Sciences, Ibn Zohr University, Morocco
[2,6]SIG, IRIT, Paul Sabatier University, France
[3,4]Department of Mathematics and IT, ENSA, Mohamed 1st University, Morocco

Article Info

Article history:

Received Oct 9, 2017
Revised Jan 30, 2018
Accepted Feb 9, 2018

Keyword:

Automated clustering
Bat algortihm
Bat-cluster
Large graphs
Swarm intelligence

ABSTRACT

Defining the correct number of clusters is one of the most fundamental tasks in graph clustering. When it comes to large graphs, this task becomes more challenging because of the lack of prior information. This paper presents an approach to solve this problem based on the Bat Algorithm, one of the most promising swarm intelligence based algorithms. We chose to call our solution, "Bat-Cluster (BC)." This approach allows an automation of graph clustering based on a balance between global and local search processes. The simulation of four benchmark graphs of different sizes shows that our proposed algorithm is efficient and can provide higher precision and exceed some best-known values.

Corresponding Author:

Zakaria Boulouard,
LavSIV,
Faculty of Sciences,
Ibn Zohr University,
BP 8106, Cite Dakhla, Agadir, Morocco.
Email: zboulouard@gmail.com

1. INTRODUCTION

Graphs enable us to visualize connected data and to rely on visual prowess to decipher valuable and important hidden knowledge, which could be used to improve the decision-making process of an organization. Visualizing large graphs based on the continuously growing amount of available data has become a very complex task and has outpaced the human's ability to process, analyze, visualize and even, understand them. Therefore, a process of reducing large graphs into smaller, more representative ones is needed. To address this challenge, graph clustering has imposed itself lately as a promising research area.

Graph clustering can be defined as the problem of collecting similar nodes into same groups called "clusters." It is a widely known technique with applications in various fields such as social media [1], Web search results optimization [2], wireless sensor networks [3] and also in biochemical neural networks [4] among others. In most cases, the number of clusters to form is already known and is given as an entry to the clustering algorithm. However, with the prevalence of Big Data, it became harder to have a prior idea on the number of clusters. This also applies to the large graph clustering where the decider's visual prowess is not sufficient enough to provide him with an approximate prior idea on the eventual number of clusters. Therefore, it became imperative to propose solutions where the clustering algorithm can automatically "guess" the correct number of clusters before proceeding with the clustering operation. This research field, called "Automatic Graph Clustering," started in the late 1990's but couldn't blossom until the late 2000's early 2010's with the introduction of the artificial intelligence concepts such as nature-inspired algorithms [5], [6].

In most of the papers in the literature related to automatic graph clustering based on nature-inspired algorithms, the main idea is to define a "Similarity Measure," then set the clusters according to it. Several papers adopted a discrete formulation of this and proposed adaptations of the basically continuous nature inspired algorithms to solve it.

In the present work, we will proceed differently since we will adapt the graph clustering problem itself so it can be represented as a continuous problem. This adaptation will be using "Bat-Cluster," a combination between "FFDP," a large graph visualization algorithm we developed by our team [7], and "Bat Algorithm," a nature inspired optimization algorithm developed by Xin-She Yang [8] based on the behavior of bats. FFDP will set an equilibrium positioning of the large graph; then it will provide the nodes final positions as a vector of coordinates. Bat algorithm will take this vector into consideration and try to find the best clustering configuration possible.

After reviewing the related works in Section 2, we will describe, in Section 3, the similarity measure we used as an objective function and we will describe how it will be optimized by the "Bat-Cluster" (BC) algorithm.

The testing and results of the clustering provided by "Bat-Cluster" compared with other well-known solutions, such as PSO, Differential Evolution and Ant Colony Optimization, will be discussed in Section 4.

Section 5 concludes the paper and presents an idea of our future works.

2. RELATED WORKS

In this section, we will explore some of the most important nature inspired solutions used to answer the issue of automated graph clustering before moving to introducing Bat-Cluster in Section 3.

2.1. Particle Swarm Optimization

The literature contains several approaches to using PSO in graph clustering, often referred to as "Community Detection." Most of these approaches are based on the idea of adapting the PSO, an algorithm originally designed to solve continuous optimization problems so that it would be able to solve discrete problems. Cai et al. proposed in [9] and [10] an alteration of the definition of the position and the velocity terms where the position vector represents a partition of a signed network and the velocity represents an eventual permutation of the partition. Suganthi and Rajagopalan [11] have applied PSO in its continuous state, but they suggested using a multiple population swarm instead of using the standard PSO with one population. Rejina Parvin and Vasanthanayaki [12] used PSO to prevent residual nodes in wireless sensor networks (nodes that don't belong to any cluster). Their idea has been applied to optimize energy consumption, throughput, packet delivery ratio, and network lifetime of the wireless sensor networks.

2.2. Ant Colony Optimization

Mandala et al. [13] proposed an ACO based technique for graph clustering and applied it in detecting customer communities in the e-marketing field. Ji et al. [14] suggested a solution for the problem of complex community detection in large graphs based on the strategy of ant pheromone diffusion and update to search for an optimal graph partitioning. Zhou et al. [15] followed a similar process, but they took the overlapping issue of the large communities into consideration. Moradi and Rostami [16] used ACO along with feature selection to define clusters of features. Gao et al. [17] proposed a combination between ACO and K-Means as a solution to the dynamic location routing problem. K-Means is used to define the location of depots (cluster centers) while ACO is utilized to handle the VRP in dynamic environments.

2.3. Differential Evolution

Paterlini et al. [18] proposed a direct application of DE to solve the problem of graph partitioning and a comparative study with the Genetic Algorithm (GA) showed that DE was more efficient. Cai et al. [19] proposed an adaptation of DE inspired by the imitation of the phenomenon of social learning in animal societies. They improved the traditional DE by introducing the strategic ASL selection. It allows the algorithm to rely on the information extracted from the neighborhood relationships of its population individuals to guide the selection of the eligible parents for the crossover. Hybridization attempts of DE with other algorithms can be found in recent literature. For instance, Zorarpaci and Özil [20] suggested a combination between DE and the Artificial Bee Colony algorithm and applied it to solve the problem of feature selection.

3. PROPOSED SOLUTION : "BAT-CLUSTER"

3.1. Objective Function

The objective function for the algorithm is the quality measure that will help it decide what clustering configuration is the best. Nanda and Panda [21] provided a list of several clustering quality metrics available in the literature. What we want is a clustering able to highlight, on the one hand, the closeness between similar nodes, and on the other hand, the separation between different nodes. Therefore, the distance should have a fundamental role in choosing our quality metric. However, relying on the distance from the cluster center alone as in the traditional K-Means, or the distance between cluster centers may not be sufficient.

We need a metric able to provide a combination of these two metrics so that it would assure that the similar nodes are close to each other and far from the nodes that are different from them.

One of the most popular metrics in the literature is called "DBIndex" [22]. It was developed by Davies and Bouldin, and it provides a ratio between the intra-cluster distance (the distance between the nodes in the same cluster) and the inter-cluster distance (the distance between the centers of each cluster).

DBIndex is defined as:

$$DB = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i \tag{1}$$

Where:

$$R_i = \max_{j=1...n_c, i\neq j}(R_{ij}),\ i = 1...n_c$$

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

$$d_{ij} = d(g_i, g_j),\ s_i = \frac{1}{\|c_i\|}\sum_{x\in c_i} d(x, g_i)$$

$d(x,y)$ is the Euclidian distance between x and y.

c_i is the cluster i.

g_i is the center of the cluster i.

$\|c_i\|$ is the norm of c_i.

According to Davies and Bouldin [22], a correct clustering minimizes the DBIndex as depicted in Equation (1). That being said, the objective function for our clustering algorithm should be:

$$\min DB = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i \tag{2}$$

To solve Equation (2), we propose a hybridization of the standard Bat Algorithm by Yang [8] with the FFDP algorithm that we developed in a previous work [7]. We chose to call this hybridization "Bat-Cluster," or BC.

3.2. Bat-Cluster

Bat-Cluster, or BC, is a combination of two algorithms, FFDP and Bat Algorithm. FFDP will run first to set an optimized equilibrium positioning of the nodes of the graph. These node positions will then be assigned to the Bat Algorithm.

BA will start by generating a population of bats. Each of these bats will have its own initial loudness, pulse rate, position and velocity. The initial bats positions represent the initial cluster centers. When the algorithm starts running, each bat will be assigned to a cluster center location. For each cluster

center, the algorithm will calculate the mean value of the closest nodes to it. The cluster center's position is then updated, and the objective function is calculated as in the Equation (2). If the value of the objective function has converged, we return the cluster center locations; otherwise, we reassign each bat to the corresponding cluster center once again.

If the random value $rand$ is greater than the bats pulse rate, the algorithm selects a solution among the best solutions and generates a local solution around the selected best solution. If the random value $rand$ is smaller than the loudness A_i, and the value of the objective function for the current bat position is better (smaller in our case) than the value of the best solution found so far, $f(x_i) \prec f(x^*)$, the new solution is accepted, the bats pulse rate is increased, and the loudness is decreased. The solutions found are sorted, and the current best solution is stored.

The algorithm keeps running until the stop criterion is respected. In our case, the algorithm should stop if the iteration number t becomes equal to the maximum number of iterations M.

The pseudo code of the Bat-Cluster algorithm will be then described as follows:

Algorithm BatCluster(G, X, tol, K, M, N, A^0, r^0)

---

input: Graph G, Nodes initial positions X, tolerance tol, nominal edge length K, Maximum iterations number M, Bats total population N, initial loudness A^0 and initial pulse rate r^0

// Set the graph's positioning

$coords = FFDP(G, X, tol, K)$ //coords is 2-D or 3-D vector

/*Initialize the bats positions x_i, velocities v_i and frequencies f_i */

For all bats i **do** {

$A_i = A_0$

$r_i = r_0$

$f_i = 0$

$v_i = 0$

$x_i = rand$

/* Calculate the solution of the objective function for the bat i according to the equation (2) */

$DB(x_i, coords)$

}

// Select the best solution

While ($t \prec M$) {

/* Generate a new solution by adjusting the frequencies of the bats according to the equation (4), updating the velocities and the positions of the bats according to the equations (5) and (6) */

For all bats i **do** {

$f_i = f_{\min} + (f_{\max} - f_{\min})\beta$

$v_i^{t+1} = v_i^t + (x_i^t + x^*) f_i$

$x_i^{t+1} = x_i^t + v_i^{t+1}$

If ($rand \succ r_i$) {

/* Generate a new solution around the best solution according to the equation (7) */

$x_{new} = x_{old} + \varepsilon A^t$

}

}

If ($rand \prec A_i$ *and* $DB(x_i, coords) \prec DB(x^*, coords)$) {

//Accept the new solutions

$x^* = x_i$

/* Update A_i and r_i according to equations (8) and (9) */

$A_i^{t+1} = \alpha A_i^t$

$$r_i^{t+1} = r_i^0 \left[1 - e^{(-\gamma t)} \right]$$

}
}
// Select the current best solution
$t = t + 1$
}

4. TESTING AND RESULTS

4.1. Testing Environment

For simulation purposes, we ran the Bat-Cluster algorithm in a computer with the following technical characteristics:

a. Environment: Java 8 + Matlab R2017a
b. OS: Windows 7
c. CPU: Intel i5 2450M 2.5Ghz
d. RAM: 4Gb

To test the performances of BC, we put it in comparison with three distinguished algorithms:

e. Particle Swarm Optimization
f. Ant Colony Optimization
g. Differential Evolution

We will use the continuous aspect of all these algorithms, and the function to optimize will be the DBIndex as depicted in the Equation (2). This approach will enable us to compare the performances of these algorithms on an equal foot.

4.2. Benchmark Graphs

The graphs that we will use in our tests are three benchmark graphs of different sizes and come from different domains. These graphs are available in the Gephi standard dataset accessible in the following link:

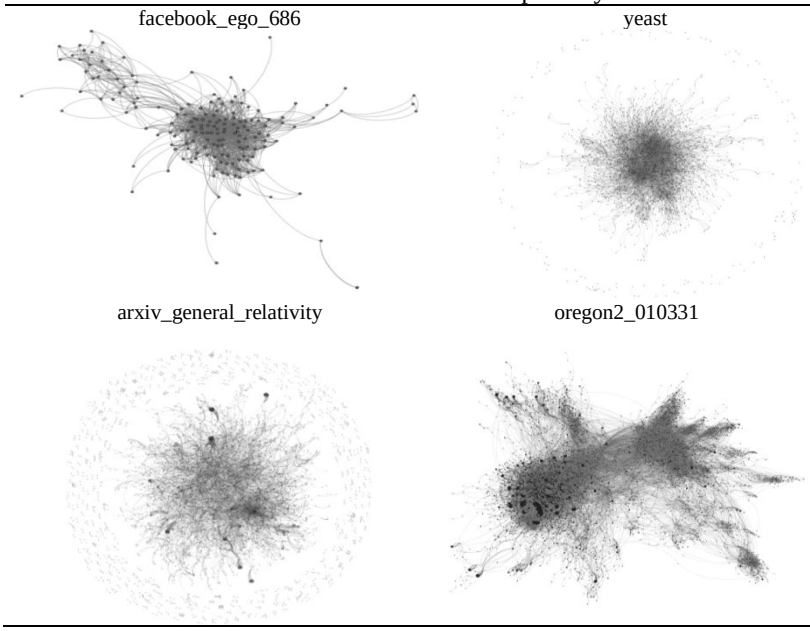
https://github.com/medialab/benchmarkForceAtlas2/blob/master/dataset.zip (last checked: December 21th, 2017).

The 4 graphs are:

a. facebook_ego_686 (168 nodes and 1656 links)
b. yeast (2361 nodes and 7182 links)
c. arxiv_general_relativity (5242 nodes and 28980 links)
d. oregon2_010331 (10900 nodes and 31180 links)

Table 1 displays the layouts of the 4 benchmark graphs.

Table 1. The Benchmark Graphs Layouts



4.3. Parameters Setting

Defining the correct parameters for a nature inspired algorithm, in general, requires rigorous prior testing. The same goes for BC and all of the algorithms that we will test it against.

After several experiments, the parameters we found able to answer our needs correctly are the following:

a. Bat-Cluster: Maximum Iterations Number $M = 200$, Bat Population Size $N = 50$, Initial Loudness $A_0 = 1.1$, Initial Pulse Rate $r_0 = 0.5$

b. Particle Swarm Optimization: Maximum Iterations Number $M = 200$, Particle Population Size $N = 50$, Inertia Weight $w = 1$, Inertia Weight Damping Ratio $wdamp = 0.99$, Personal Learning Coefficient $c_1 = 1.5$, Global Learning Coefficient $c_2 = 2$

c. Ant Colony Optimization: Maximum Iterations Number $M = 200$, Ant Population Size $N = 10$, Sample Size $nSample = 40$, Intensification Factor $q = 0.5$, Deviation Distance Ratio $\zeta = 1$

d. Differential Evolution: Maximum Iterations Number $M = 200$, Individual Population Size $N = 50$, Crossover Probability $pCR = 0.2$

4.3. Experimental Results

The Table 2 to Table 5 show the performances of the Bat-Cluster compared with each of the other aforesaid algorithms on the four benchmark graphs.

Table 2. facebook_ego_686

Algorithm	Number of Clusters	DBIndex Values
BC	4	0,72657
PSO	3	0,72731
ACO	3	0,80209
DE	3	0,73884

In “facebook_ego_686”, Bat-Cluster provided the smallest optimal value for the DBIndex, closely seconded by PSO. Yet, BC was the only algorithm able to provide three clusters while the other algorithms provided only 3 clusters.

Table 3. yeast

Algorithm	Number of Clusters	DBIndex Values
BC	4	0,69332
PSO	5	0,69423
ACO	3	0,81152
DE	3	0,71968

The results in the “yeast” graph can be debatable at first. Indeed, based on the DBIndex alone, we will say that BC was the best, but seeing that PSO was able to provide more clusters can open the possibility that PSO may be able to find a better value for the DBIndex. However, according to the evolution of the best values as depicted in Figure 1, we will see that PSO started stagnating after the iteration 100 in a DBIndex value higher than the one provided by BC. This concludes to the fact that having 5 clusters may not be the best clustering scenario.

Table 4. arxiv_general_relativity

Algorithm	Number of Clusters	DBIndex Values
BC	6	0,74797
PSO	4	0,7558
ACO	3	0,81471
DE	3	0,76049

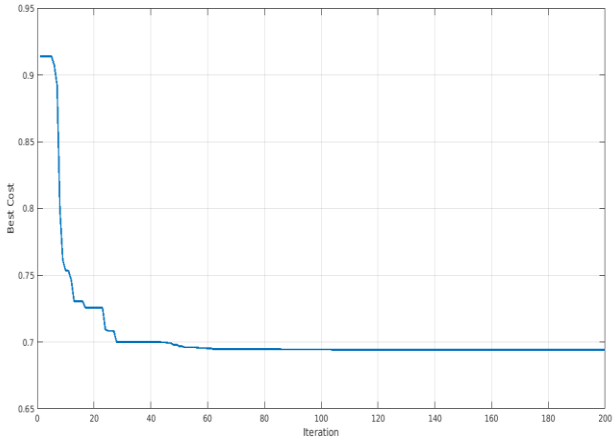


Figure 1. The evolution of the best DBIndex values provided by PSO for the "yeast" graph

In the "arxiv_general_relativity" graph, BC gave the smallest values of DBIndex and much more clusters (6 against 4 provided by the first runner-up PSO).

Table 5. oregon2_010331

Algorithm	Number of Clusters	DBIndex Values
BC	3	0,58093
PSO	3	0,58101
ACO	2	0,71037
DE	2	0,62902

Regarding the "oregon2_010331" graph, BC and PSO were able to provide 3 clusters, while the other two could only provide 2 clusters. The DBIndex values of BC and PSO were very close, with a small advantage for BC.

Overall, Bat-Cluster was able to provide the best values of DBIndex on all the benchmark graphs. Being closely seconded by PSO shows the ability of the Swarm Optimization algorithms to tackle this kind of problems. However, the results provided by ACO were poorer than expected. When we look at the evolution of the best value provided by ACO on "facebook_ego_686," as in Figure 2, for example, we can see that the algorithm kept finding better values. This can be explained by the fact that the configuration we gave to ACO may probably not be the best.

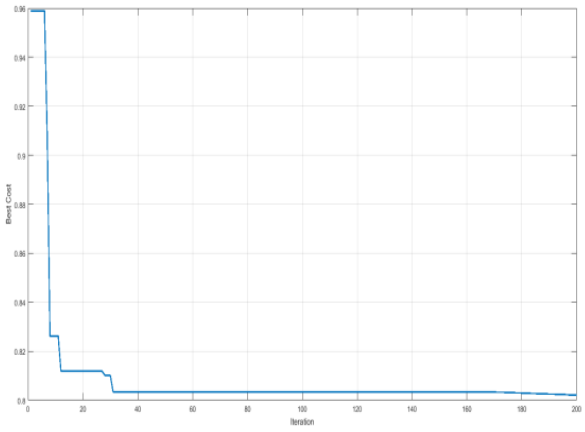


Figure 2. The evolution of the best DBIndex values provided by ACO for the "facebook_ego_686" graph

5. CONCLUSION

This paper presented the Bat-Cluster (BC) algorithm. It is a combination of the FFDP algorithm developed by our team [7] and the Bat Algorithm developed by Xin-She Yang[8]. BC is an algorithm designed to answer the need for automated large graph clustering. In contrast with several clustering algorithms available in the literature, BC was able to translate the automated large graph clustering issue into a continuous problem, while the other solutions tend to formulate it as a discrete problem. The idea here was to run a large graph layout algorithm, the FFDP, and make it provide the coordinates of the equilibrium positions of the graph's nodes. Having these coordinates enabled us to translate the graph to a standard real valued vector easily solvable with the continuous version of the Bat Algorithm. The quality metric we used to measure the quality of our clustering was the DBIndex by Davies and Bouldin [22]. The Bat-Cluster algorithm was tested on four benchmark graphs of different sizes and from different domains. BC proved to be a good alternative solution to solve the automated large graph clustering problem when compared to algorithms considered among the best in the literature.

The Bat-Cluster algorithm will be integrated into XEWGraph [23], the large graph visualization service of the Competitive Intelligence tool Xplor EveryWhere [24]. Coupled with the out of the box categorization provided by XEWGraph's hypergraph approach, BC will enable the user to have large graphs clustered and expanded on demand for both the web and the mobile oriented interfaces of XEWGraph.

REFERENCES

[1] C. Virmani, A. Pillai, and D. Juneja, "Clustering in Aggregated User Profiles across Multiple Social Networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 6, pp. 3692–3699, Dec. 2017.

[2] S. Jinarat, C. Haruechaiyasak, and A. Rungsawang, "Graph-Based Concept Clustering for Web Search Results," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, no. 6, pp. 1536–1544, Dec. 2015.

[3] A. Mahboub, M. Arioua, and E. M. En-Naimi, "Energy-Efficient Hybrid K-Means Algorithm for Clustered Wireless Sensor Networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 4, pp. 2054–2060, Aug. 2017.

[4] H. Zhou and R. Lipowsky, *"Dynamic pattern evolution on scale-free networks,"* Proceedings of the National Academy of Sciences of the United States of America, vol. 102, no. 29, pp. 10052–7, Jul. 2005.

[5] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 39, no. 2, pp. 133–155, 2009.

[6] D. Camacho, *"Bio-inspired Clustering: basic features and future trends in the era of Big Data,"* in Cybernetics IEEE Conf on, 2015.

[7] Z. Boulouard, L. Koutti, A. El Haddadi, and B. Dousset, "'Forced' Force Directed Placement: a New Algorithm for Large Graph Visualization," *International Review on Computers and Software (IRECOS)*, vol. 12, no. 2, pp. 75–83, Mar. 2017.

[8] X.-S. Yang, *"A New Metaheuristic Bat-Inspired Algorithm,"* in Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 2010th ed., vol. 284, J. R. Gonzalez, D. A. Pelta, C. Cruz, T. German, and N. Krasnogor, Eds. Granada: Springer Springer, 2010, pp. 65–74.

[9] Q. Cai, M. Gong, B. Shen, L. Ma, and L. Jiao, "Discrete particle swarm optimization for identifying community structures in signed social networks," *Neural Networks*, vol. 58, pp. 4–13, Oct. 2014.

[10] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, and L. Jiao, "Greedy discrete particle swarm optimization for large-scale social network clustering," *Information Sciences*, vol. 316, pp. 503–516, Sep. 2015.

[11] S. Suganthi and S. P. Rajagopalan, "Multi-Swarm Particle Swarm Optimization for Energy-Effective Clustering in Wireless Sensor Networks," *Wireless Personal Communications*, vol. 94, no. 4, pp. 2487–2497, Jun. 2017.

[12] J. Rejina Parvin and C. Vasanthanayaki, "Particle Swarm Optimization-Based Clustering by Preventing Residual Nodes in Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4264–4274, Aug. 2015.

[13] S. R. Mandala, S. R. T. Kumara, C. R. Rao, and R. Albert, "Clustering social networks using ant colony optimization," *Operational Research*, vol. 13, no. 1, pp. 47–65, Apr. 2013.

[14] J. Ji, X. Song, C. Liu, and X. Zhang, "Ant colony clustering with fitness perception and pheromone diffusion for community detection in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 15, pp. 3260–3272, Aug. 2013.

[15] X. Zhou, Y. Liu, J. Zhang, T. Liu, and D. Zhang, "An ant colony based algorithm for overlapping community detection in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 427, pp. 289–301, Jun. 2015.

[16] P. Moradi and M. Rostami, "Integration of graph clustering with ant colony optimization for feature selection," *Knowledge-Based Systems*, vol. 84, pp. 144–161, 2015.

[17] S. Gao, Y. Wang, J. Cheng, Y. Inazumi, and Z. Tang, "Ant colony optimization with clustering for solving the dynamic location routing problem," *Applied Mathematics and Computation*, vol. 285, pp. 149–173, Jul. 2016.

[18] S. Paterlini and T. Krink, "Differential evolution and particle swarm optimisation in partitional clustering," *Computational Statistics & Data Analysis*, vol. 50, no. 5, pp. 1220–1247, Mar. 2006.

[19] Y. Cai, J. Liao, T. Wang, Y. Chen, and H. Tian, "Social learning differential evolution," *Information Sciences*, Oct. 2016.

[20] E. Zorarpacı and S. A. Özel, "A hybrid approach of differential evolution and artificial bee colony for feature selection," *Expert Systems with Applications*, vol. 62, pp. 91–103, Nov. 2016.
[21] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary Computation*, vol. 16, pp. 1–18, 2014.
[22] D. L. Davies and D. W. Bouldin, "DBIndex : A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.
[23] Z. Boulouard, L. Koutti, A. E. Haddadi, A. E. Haddadi, and A. Fennan, "XEWGraph : A tool for visualization and analysis of hypergraphs for a competitive intelligence system," in *SIIE 2015 - 6th International Conference on "Information Systems and Economic Intelligence,"* 2015, pp. 66–70.
[24] A. El Haddadi, "Fouille multidimensionnelle sur les données textuelles visant à extraire les réseaux sociaux et sémantiques pour leur exploitation via la téléphonie mobile," Université de Toulouse III, Paul Sabatier, 2011.