A Genetic Algorithm with Local Search Based on Label Propagation for Detecting Dynamic Communities

Conference Paper *in* Studies in Computational Intelligence · October 2018

DOI: 10.1007/978-3-319-99626-4_28

3 authors, including:

Ángel Panizo Lledot
Universidad Politécnica de Madrid
16 PUBLICATIONS **22** CITATIONS

SEE PROFILE

David Camacho
Universidad Politécnica de Madrid
317 PUBLICATIONS **2,771** CITATIONS

SEE PROFILE

A Genetic Algorithm with local search based on Label Propagation for detecting dynamic communities

A. PANIZO[1], G. BELLO-ORGAZ[1], D. CAMACHO[1],

1 Computer Science Department, Universidad Autónoma de Madrid, Madrid, Spain * {angel.panizo, gema.bello, alfonso.ortega,

david.camacho,}@uam.es

Acknowledgments

This work has been co-funded by the following research projects: EphemeCH (TIN2014-56494-C4-4-P) and DeepBio (TIN2017-85727-C4-3-P) projects (Spanish Ministry of Economy and Competitivity, under the European Regional Development Fund FEDER).

Abstract

The interest in community detection problems on networks that evolves over time have experienced an increasing attention over the last years. Genetic Algorithms, and other bio-inspired methods, have been successfully applied to tackle the community finding problem in static networks. However, few research works have been done related to the improvement of these algorithms for temporal domains. This paper is focused on the design, implementation, and empirical analysis of a new Genetic Algorithm pair with a local search operator based on Label Propagation to identify communities on dynamic networks.

1 Introduction

The analysis of complex networks is a relevant research topic because many real world problems can be modeled as a complex network. Identifying communities within these networks have been applied to disciplines such as marketing [2, 5], public healthcare [4], cybercrime [14, 15, 19] or ego-based social network analysis [10, 16]. There are several definitions of a community, but no consensus has been reached between the scientific community about what is the correct one. One commonly accepted definition for a community is: a set of nodes that have stronger interactions between them than among the rest of nodes of the network. Two types of communities can be defined. The first ones are composed of nodes that can belong to several communities at the same time, and are called *overlapping communities*. The second ones are composed of nodes that can only belong simultaneously to a single community, and are called *non-overlapping communities*.

Dynamic networks (networks that change over time) has received a great deal of attention lately, mainly due to the temporal nature of real-world networks such as social networks or mobile communications. A dynamic network is usually modeled using a sequence of snapshots, where each snapshot represents the network at a given point in time. The communities found on these networks are called *dynamic communities*. To identify these type of communities not only we need to partition each snapshot but also we need to track the partitions between

snapshots. R.Alhajj [1] defines a *static community* as a set of nodes and edges among them, that do not change in time. This author also introduces two possible definitions of a *dynamic community*. The first one as a sequence of *static communities*, one for each snapshot in the dynamic network. And a second one, as an initial *static community* and a series of modifications of it over time.

Dynamic community finding methods can be split in two groups. A first group that contains the methods that separate the community detection step from the temporal analysis, i.e first, the communities are extracted and then the structural differences over time are analyzed; and a second one that follows the *evolutionary clustering* framework [8], to simultaneously optimize the two conflicting criteria community detection and temporal analysis of the communities at each snapshot. These methods assume that abrupt changes in the communities in a short time period do not happen, thus they look for *temporal smoothness* in the dynamic communities. For clarity purposes, we will call the methods in the first group *non temporal smooth* methods, and the second ones *temporal smooth* methods.

Genetic Algorithms (GAs) are a stochastic meta-heuristic inspired in the biological principles of Darwinian theory of evolution and Mendel's genetics. GAs are an effective method to solve combinatorial optimization problems, and have been used before to solve both the static [2, 3, 6] and the dynamic [9] community identification problem. GAs for static community detection have been paired with *local search strategies* to improve performance. The *local search strategies* are applied mostly during the *population initialization* [20] or during the *mutation process* [17]. In this paper, we introduce a GA with a *local search strategy* based on *Label Propagation* [24] that detects non-overlapping dynamic communities following a *non temporal smooth* approach. This work is focused on the community detection step, leaving the subsequent temporal analysis out of the scope. In *Label Propagation* each node starts in its own community and following an iterative process each node updates its community to match the community of the majority of its neighbors . The main idea behind our *local search strategy* is that consecutive snapshots in a dynamic network have a similar topology, so the results obtained for one snapshot can be reused by the next one. The proposed method run the GAs for each snapshot on a dynamic network to identify its community. The initial population of each GA is constructed by applying the update method of *Label Propagation* over the result population of the previous snapshot. Finally, an experimental phase using a real world dataset has been carried out in order to test the impact of the *local search strategy* when pair with a common GAs.

The main contribution of this paper is the introduction and testing of a new local search operator capable of extracting information from previously found solutions in order to boost the performance of a GAs that detects dynamic communities in dynamic networks modeled as a sequence of snapshots.

The rest of the article is structured as follows: section 2 reviews other related methods for community finding. Section 3 describes the dynamic community detection problem and the proposed algorithm. Section 4 presents the procedure followed to test the algorithm and discusses the experimental results. Finally, the last section draws some conclusions and future research lines of the work.

2 Related Work

Two problems have to be solved for detecting dynamic communities: grouping nodes for each snapshot, and tracking these groups over time. Some works are only focused on how to track groups over time, and assume that the groups have already been discovered [7, 11]. GED [7] defines seven independent events (Continuing, Shrinking, Growing, Splitting, Merging, Dissolving and Forming) that could change a group state, and a new measure called *inclusion* is used to decide which events occurs. Other work following this approach has been proposed by Greene et al. [11], that also defines seven independent events, very similar to the

previous ones defined by GED. But in this work, the *Jaccard coefficient for binary sets* is used to decide which event occurs.

On the other hand, some research works are focused on solving both problems, detection and tracking of the groups or communities. Two of the most well-known methods related to this approach are *GraphScope* [25] and *FacetNet* [18]. The first one is a *non temporal smooth* method based on the Minimum Description Length (MDL) principle, and it is able to group nodes and track them in a streaming way. It reorganizes snapshots into segments, given a new incoming snapshot, it is combined with the current segment if there is a storage benefit, otherwise, the current segment is closed, and a new one is started with the new snapshot. A change of segment means that an event has occurred on the network. *FacetNet* is a *temporal smooth* method that uses probabilistic learning in order to identify dynamic communities. *FacetNet* follows Chakrabarti et al. [8] equation where a parameter α governs the importance between the *snapshot cost* and the *temporal cost*. To solve this equation they presents a low time complexity iterative algorithm which is guaranteed to converge to an optimal solution.

In particular, regarding GAs approaches to detect dynamic communities, Folino and Pizzuti present the DYMONGA [9] algorithm. This is a *temporal smooth* method based on a multi-objective approach, that uses the *modularity* to measure the quality of the communities, and the Normalized Mutual Information (NMI) to measure the change between two consecutive snapshots. DYMONGA is only focused on grouping the nodes in each snapshot. Kim et al. [12] also propose a multi-objective GA that uses immigrant schemes to track the best grouping of the network over time. In contrast to DYMONGA, the method proposed by Kim et al. returns the partition that best suits the last snapshot, taking into account all the previous snapshots, instead of returning the partitions for each snapshot.

Finally, there are several works that use GAs with local search in order to improve the performance of the method for detecting static communities. For example, Li et al. [17] introduce a new mutation operator, and apply a local search operator based on the small-world phenomena. The mutation operator creates n copies of the individual to be mutated. For each copy, several gens are randomly selected. For each selected gen, all the neighbors related to that gen move to the same community of the gen. The individual to be mutated is substituted by the copy with higher fitness. Wang et al. [26] proposed a modify crossover operator more suitable for community detection, and an heuristic mutation operator based on local modularity. When the mutation operator changes a gene, the node related to that gene is moved to the same community of the neighbor that has more edges in the local community. Finally, Pizzuti [23] applies local search after the GA has finished to boost community quality. The local search performs a greedy search over the best individual found by the GAs and tries to move each node of the network to the community of the neighbor that gets higher modularity after receiving it.

All the local search methods described in the paragraph above works on static environments, environments that never change. Their objective is taking an already valid solution and improving it if possible. On the contrary, our local search method works on dynamic environments, environments that change in time. In a dynamic environment is possible that a valid solution became invalid due to some changes in the environment, some nodes or edges have been removed or added to the graph. The objective of our local search operator is not improving a valid solution, but transforming an invalid solution, that have stopped being valid due to change, into a valid one maintaining the highest possible quality.

3 Algorithm Description

3.1 Problem Formulation

Let $\aleph = \{G^0..G^n\}$ be a dynamic network with n snapshots sorted in chronological order. Each snapshot G^t is modeled as a graph $G^t(V^t, E^t)$ where V^t is a set of vertices and E^t is a set of links, called edges, that connects two elements of V^t at time t. Our goal is to group the vertices in each G^t in such a way that vertices in the same group have more links among them than with the rest of vertices of G^t, and each vertex in G^t only belongs to one single group.

3.2 Genetic algorithm with local search

The proposed algorithm is shown in *Algorithm* 1. It uses an elitist GA for detecting communities in each snapshot(G^t) of the dynamic network $\aleph$ (lines 1-13). The method evolves a random population for the first snapshot (G^0). The population is generated following an heuristic that will be described later. However, for the rest of snapshots ($G^t|t > 0$), instead of evolving a random population, it evolves the population generated by applying the local search operator to the population returned by the previous GA run. For each run of the GA, the best individual (C_{last}) of the last population generated is selected as final result (line 13), which will be used to apply the local search operator to the next run.

Each individual represents a possible grouping of the current snapshot(G^t) using the locus-based adjacency representation [22]. For the first snapshot (G^0), a random population, generated according to this encoding, is used as initial population (line 7). The random population is generated following the next heuristic: each random individual of the population is generated by filling each position in the chromosome with a random neighbor, in G^0, of the node related with that position. On the other hand, for the rest of snapshots($\{G^t|t > 0\}$), the individuals of the initial population are generated by applying the *labelLocalSearch* function to the individuals returned by the last run of the GA (lines 9-11). This function (lines 16-23) changes those genes inside the chromosome with values that refers to a node that no longer is a neighbor of that gene. To select a new value for it, first we find all the neighbors of the node represented by the gene in the actual snapshot(G^t) (line 18). Then, using the best solution (C_{last}) for the past snapshot (G^{t-1}), the algorithm selects the community of C_{last} which contains the higher number of the neighbors obtained previously (line 20). Finally, the value of the gene is swap with a random value among the neighbors that also belong to the mentioned community (line 22).

Once the initial population (random or not) has been created, it evolves using an elitist GA (line 12) with the next functions: *Two Points Crossover*, *Tournament Selection*, *Modularity* [21] as *Fitness function*. Finally, a specific mutation operator has been implemented, that randomly selects a series of genes, following the normal distribution, to change them with a random neighbor of the corresponding node. After testing with several combinations of the most common functions used in the literature, we have selected the combination described above.

4 Experimentation

4.1 Dataset Description

In this section we study the effectiveness of our method using the *Enron* [13] dataset. This dataset consists of emails sent by the workers of the Enron corporation between 1999 and 2003. In this dataset each node is an Enron worker and an edge connecting two nodes means that an email has been exchanged between those two workers. This communication by email has associated the date when it happened. So, a dynamic network can be generated, splinting the Enron

Algorithm 1 GA with local search for detecting dynamic communities

```
1: function LocalSearchGA(ℵ)
2:     C_dynamic ← ∅
3:     P_last ← null
4:     C_last ← null
5:     for all G^t ∈ ℵ do
6:         if t = 0 then
7:             P_initial ← makeRandomPopulation(G^t, POPSIZE)
8:         else
9:             P_initial ← ∅
10:            for all individual ∈ P_last do
11:                P_initial ← P_initial ∪ labelLocalSearch(individual, G^t, C_last)
12:        P_last ← elitistGA(P_initial, G^t, ELITISM, CRXRATE, MUTRATE, MAXGEN, N
13:        C_last ← getBestSolution(P_last)
14:        C_dynamic ← C_dynamic ∪ C_last
15:    return C_dynamic
16: function labelLocalSearch(individual, G^t, C_last)
17:    for all i ∈ [1, size(individual)] do
18:        neighbors ← getNeighbors(i, G^t)
19:        if individual[i] ∉ neighbors then
20:            community ← getCommunityOfMost(neighbors, C_last)
21:            validNeighbors ← getNodesInCommunity(neighbors, community)
22:            individual[i] ← selectRandom(validNeighbors)
23:    return individual
```

network in several snapshots using these dates. In this case, each snapshot has all the emails sent over fifteen days, the weekends have been ignored because very little activity occurred on them. The snapshots are fifteen days long because shorter snapshots makes the network too dynamic to be analyzed. A total of 8 snapshot have been created, from the 1st of January of 2000 through the 30th of April of the same year. The number of nodes and edges of each snapshot in the network can be seen in the figure 1.

4.2 Experimental Results

In order to validate the effectiveness of our method we have compared it against an elitist GA without local search (evolves a new random population for every snapshot). For this comparison, the fitness of the best individual of every generation achieved by both methods is used. This allows us to check which method reaches a higher fitness value, and also how many generations are required to achieve it. Both GAs, the one with local search and the one without it, uses the same set-up. The parameter of the set-up can bee seen on Table 1. The different parameters of the algorithms have been tuned up experimentally. These settings have been used during all the experimentation phase.

Each experiment have been performed 30 times, and the median of all executions can be seen in the Figure 2. Analyzing the results we can conclude that the elitist GA with local search achieves better results in almost every snapshot, excluding the *snapshot 3* (Figure **??**). The *snapshot 0* (Figure **??**) is a special case due to there is no previous solutions to perform the local search, and then both elitist GAs generate a random initial population. Therefore, in this particular case both GAs achieve similar results. For the rest of snapshots, beside the *snapshot 3*, the fitness value of the best individual is better during all the generations of the GAs. In the *snapshot 3* the population of the GA with local search starts being

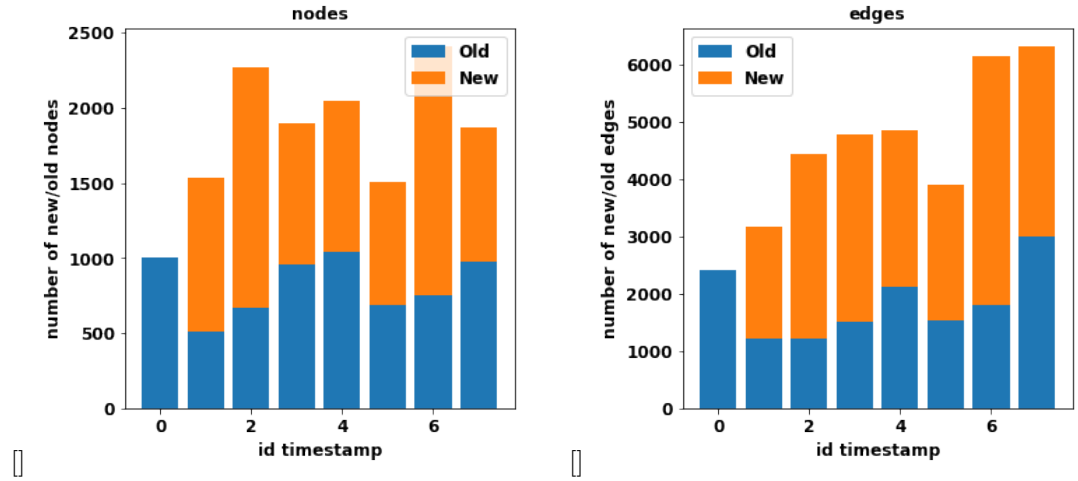

Figure 1. Number of nodes and edges for each snapshot in the dynamic network. The blue part of the bars indicates the number of elements that existed on the previous snapshot, an the orange part represents the number of elements that are new in that particular snapshot.

Table 1. Genetic parameters of both GAs (with and without local search).

Parameter	Description	Value
$POPSIZE$	Population size	300
$ELITISM$	Number of individuals to select for the next generation	10%
$MAXGEN$	Maximum number of generations	500
$NCONV$	Number of generations with same best fitness (+/-0.01)	10
$CRXRATE$	Crossover probability	1.0
$MUTRATE$	Mutation probability	0.1

better but after the 15th generation the GA without local search surpasses it and achieves a better final result. In our opinion the main drawback of the local search method lies here: although the population of the different snapshots starts with better fitness values, the convergence speed of the algorithm tends to decrease. The local search method creates populations with lower diversity and reduces the exploration capacities of the GA. This makes the GA more likely to get trapped in a local optima.

5 Conclusions

In this paper we present a new evolutionary approach with a local search operator to detect communities in dynamic networks. The method executes an elitist GA to detect the communities in each snapshot. However, in order to improve the quality of the solutions, a local search operator is applied. This new operator is based on the *Label Propagation* method, and it uses the information of the last population generated by the GA at the previous snapshot, to generate the initial population of the current snapshot. This way, the knowledge generated in past populations can be reused to improve future ones. The experiments performed show that the method improves the global quality of the solutions at the expense of exploration capabilities in general terms. This could make the method more prone to getting trapped in a local optima, although in most of the tested cases this fact does not occur. Future research will focus on enhancing the exploration capabilities of the method. For this purpose, we will try to generate more diverse initial populations by using different local search operators over the same population. Besides, comparing our method with other well-know meta-heuristics found in the literature to determine if it is promising or not.

References

1. R. Alhajj and J. Rokne. *Encyclopedia of social network analysis and mining*. Springer Publishing Company, Incorporated, 2014.
2. G. Bello, H. Menéndez, S. Okazaki, and D. Camacho. Extracting collective trends from twitter using social-based data mining. In C. Bădică, N. T. Nguyen, and M. Brezovan, editors, *Computational Collective Intelligence. Technologies and Applications*, pages 622–630, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
3. G. Bello-Orgaz and D. Camacho. Evolutionary clustering algorithm for community detection using graph-based information. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 930–937, July 2014.
4. G. Bello-Orgaz, J. Hernandez-Castro, and D. Camacho. Detecting discussion communities on vaccination in twitter. *Future Generation Computer Systems*, 66:125–136, 2016.
5. G. Bello-Orgaz, H. Menéndez, S. Okazaki, and D. Camacho. Combining social-based data mining techniques to extract collective trends from twitter. *Malaysian Journal of Computer Science*, 27(2), 2014.
6. G. Bello-Orgaz, H. D. Menéndez, and D. Camacho. Adaptive k-means algorithm for overlapped graph clustering. *International journal of neural systems*, 22(05):1250018, 2012.
7. P. Bródka, S. Saganowski, and P. Kazienko. Ged: the method for group evolution discovery in social networks. *Social Network Analysis and Mining*, 3(1):1–14, 2013.
8. D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560. ACM, 2006.

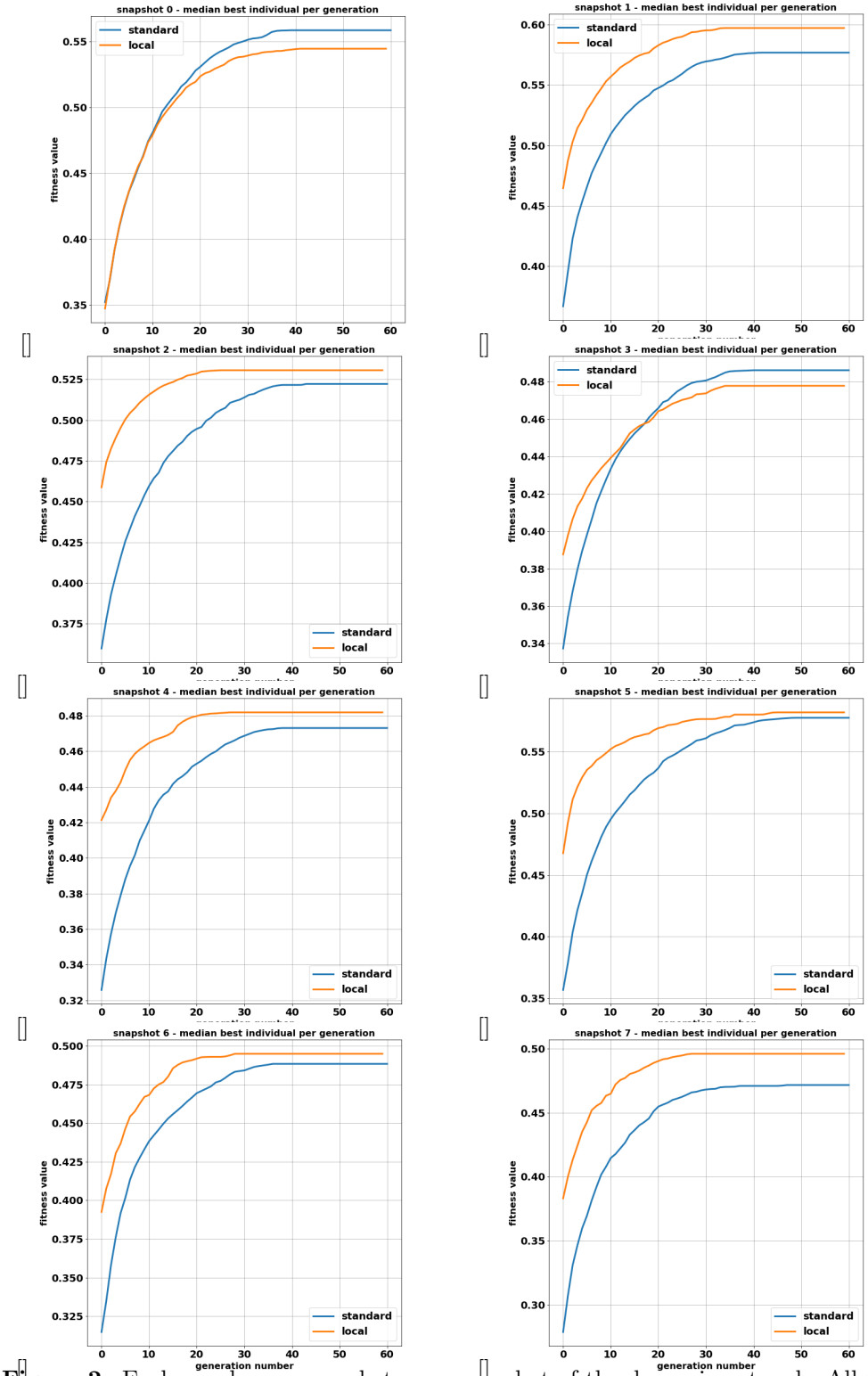

Figure 2. Each graph corresponds to one snapshot of the dynamic network. All the graph show the fitness function value of the best individual of the population over the generations.

9. F. Folino and C. Pizzuti. An evolutionary multiobjective approach for community discovery in dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1838–1852, 2014.
10. A. Gonzalez-Pardo, J. J. Jung, and D. Camacho. Aco-based clustering for ego network analysis. *Future Generation Computer Systems*, 66:160–170, 2017.
11. D. Greene, D. Doyle, and P. Cunningham. Tracking the evolution of communities in dynamic social networks. In *Advances in social networks analysis and mining (ASONAM), 2010 international conference on*, pages 176–183. IEEE, 2010.
12. K. Kim, R. I. McKay, and B.-R. Moon. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1179–1186. ACM, 2010.
13. B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.
14. R. Lara-Cabrera, A. Gonzalez-Pardo, M. Barhamgi, and D. Camacho. Extracting radicalisation behavioural patterns from social network data. In *2017 28th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 6–10, Aug 2017.
15. R. Lara-Cabrera, A. Gonzalez-Pardo, and D. Camacho. Statistical analysis of risk assessment factors and metrics to evaluate radicalisation in twitter. *Future Generation Computer Systems*, 2017.
16. R. Lara-Cabrera, A. G. Pardo, K. Benouaret, N. Faci, D. Benslimane, and D. Camacho. Measuring the radicalisation risk in social networks. *IEEE Access*, 5:10892–10900, 2017.
17. S. Li, Y. Chen, H. Du, and M. W. Feldman. A genetic algorithm with local search strategy for improved detection of community structure. *Complexity*, 15(4):53–60, 2010.
18. Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 685–694. ACM, 2008.
19. A. Malm and G. Bichler. Networks of collaborating criminals: Assessing the structural vulnerability of drug markets. *Journal of Research in Crime and Delinquency*, 48(2):271–297, 2011.
20. S. B. Mathias, V. Rosset, and M. C. Nascimento. Community detection by consensus genetic-based algorithm for directed networks. *Procedia Computer Science*, 96:90–99, 2016.
21. M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.
22. Y. Park and M. Song. A genetic algorithm for clustering problems. In *Proceedings of the third annual conference on genetic programming*, volume 1998, pages 568–575, 1998.
23. C. Pizzuti. Boosting the detection of modular community structure with genetic algorithms and local search. In *Proceedings of the 27th annual ACM symposium on applied computing*, pages 226–231. ACM, 2012.
24. U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.
25. J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM, 2007.

26. S. Wang, H. Zou, Q. Sun, X. Zhu, and F. Yang. Community detection via improved genetic algorithm in complex network. *Information Technology Journal*, 11(3):384–387, 2012.