On the Effectiveness of Evolutionary Computation for the Modularity Maximization Problem

Fernando Concatto
Laboratório de Inteligência Aplicada
Universidade do Vale do Itajaí (UNIVALI)
Caixa Postal 360 – CEP 88302-202 – Itajaí – SC – Brasil
fernandoconcatto@gmail.com

Alex Luciano Roesler Rese
Laboratório de Inteligência Aplicada
Universidade do Vale do Itajaí (UNIVALI)
Caixa Postal 360 – CEP 88302-202 – Itajaí – SC – Brasil
alexrese@outlook.com

Rafael de Santiago
Laboratório de Inteligência Aplicada
Universidade do Vale do Itajaí (UNIVALI)
Caixa Postal 360 – CEP 88302-202 – Itajaí – SC – Brasil
rsantiago@univali.br

ABSTRACT

Community detection is an important topic in many fields of science. Taking note of the pervasiveness of this problem, the research community developed several algorithms to find optimal clusterings of a network as quickly as possible, also developing a function called modularity to quantify the quality of a partition. In particular, many publications proposed the usage of a wide range of Evolutionary Computation strategies to solve the problem of modularity maximization, with varying degrees of success. However, a considerable portion of these proposals either had to be adapted to be able to solve the problem or took too much time to find good solutions, raising questions about their effectiveness in this context. This paper explores these strategies and other studies about the characteristics of modularity, arguing for strengths and weaknesses.

KEYWORDS. Evolutionary Computation. Community Detection. Modularity Maximization.

Metaheuristics, Theory and Algorithms in Graphs, Combinatorial Optimization

1. Introduction

Complex networks have become a widely discussed topic in many fields of science. Biologists have employed complex networks in the analysis of interconnected systems, such as neural and metabolic networks [Achacoso and Yamamoto, 1991; Jeong et al., 2000; Meunier et al., 2014] and in the identification of protein complexes [Nepusz et al., 2012]. In the context of social sciences, complex networks have been applied in the analysis of collaborations by scientists [Newman, 2001] and criminal organizations [Ferrara et al., 2014]. Researchers in the medical sciences have also been employing complex networks in their studies, such as in the examination of disease transmission networks [Bell et al., 1999] and in the mitigation of epidemics in a variety of situations [Bishop and Shames, 2011; Marcelino and Kaiser, 2012; Pastor-Satorras et al., 2015].

Real-world networks display a high level of organization, a characteristic that is absent from mathematical networks such as lattices and Erdős–Rényi (random) networks [Fortunato, 2010]. An important aspect of these systems is their community structure, where a community is characterized by a high connection density between its components, but few connections to elements outside of it. Bedi and Sharma [2016] states that a community can be defined as a group of individuals who are closer or interact more frequently in comparison to other individuals. The ability to identify these communities is relevant to a wide range of applications in many fields of science, such as in the functional analysis of biological networks [Radicchi et al., 2004; Fortunato, 2010; Fortunato and Hric, 2016].

In Newman and Girvan [2004], the *modularity* function was introduced as a measure of a partition. This function was widely adopted by the research community, and a significant portion of subsequent publications on the topic used modularity as an objective function [Fortunato, 2010]. The modularity function is defined as

$$Q(\mathcal{S}) = \sum_{C \in \mathcal{S}} \left[\frac{|E_C|}{m} - \left(\frac{\sum_{v \in C} d_v}{2m} \right)^2 \right], \tag{1}$$

where $\mathcal{S}$ is the set of all communities, composed of disjoint sets of nodes, $|E_C|$ is the number of connections between the members of community C, m is the amount of connections in the network and d_v represents the degree of a node v.

The modularity maximization problem has been proven to be $\mathcal{NP}$-hard by Brandes et al. [2008]. Due to this reason, heuristics are typically employed to find near-optimal partitions of a network [Clauset et al., 2004; Agarwal and Kempe, 2008; Blondel et al., 2008; Rotta and Noack, 2011; Aloise et al., 2013; Djidjev and Onus, 2013; Nascimento and Pitsoulis, 2013; Newman, 2013; Bedi and Sharma, 2016]. The heuristics CNM [Clauset et al., 2004] and Louvain [Blondel et al., 2008] are especially noteworthy, since they are scalable to networks with thousands of nodes.

Even though the modularity function Q provides an acceptable measurement of a partition, some studies have shown that it possesses certain degeneracies. In Fortunato and Barthélemy [2007], the authors demonstrated that detecting communities via modularity maximization suffers from a phenomenon called *resolution limit*, where modules smaller than a specific threshold fail to be distinguished from the network structure, even if these modules are cliques. Thus, two or more well-defined communities might be merged into the same cluster in the optimal solution, blurring the expected cluster structure identification. Another degeneracy is the existence of a larger than exponential number of high-quality but structurally different partitions, causing difficulties both for finding the globally optimal solution and for the interpretation of high-modularity partitions [Good et al., 2010; Fortunato and Hric, 2016].

In this paper, we focus our attention on the analysis of the effectiveness of Evolutionary Computation (EC) strategies in the context of the modularity maximization problem in complex networks. The motivation for this study is the number of publications about EC strategies that tend

to demand an unusual computational time to find partitions with a small gap on the modularity score when compared to the solutions obtained by the fastest known methods.

The next sections of this paper are divided as follows: Section 2 briefly explores techniques put forth by the research community for maximizing modularity; Section 3 reviews concepts of EC that are relevant for this work; Section 4 relates EC strategies to community detection, briefly describing typical methods for solving the problem; Section 5 thoroughly analyzes published approaches that rely on EC to maximize modularity, outlining their characteristics and comparing them with alternative strategies; finally, Section 6 presents our concluding thoughts and provides an outlook about the topics examined in this paper.

2. Related Works

Ever since the modularity objective function was put forward by Newman and Girvan [2004], there has been several published papers proposing methods to maximize the Q value. These methods apply different strategies to identify communities, like graph partitioning, function optimization, hierarchical clustering and mathematical programming [Fortunato and Hric, 2016; Bedi and Sharma, 2016].

In Clauset et al. [2004], a greedy heuristic was presented. It was scalable to more than hundreds of thousands of nodes. This heuristic consists of merging a pair of communities that best improves the modularity maximization objective function. One of the keys to its performance is the use of a heap data structure that quickly recovers the best merge option at each iteration. As the heuristic merges communities, the number of iterations is bound to the number of nodes. The usual starting solution is a partition composed of communities with a single node, so the number of communities is equal to the number of nodes in this solution.

Blondel et al. [2008] developed a local search that is known in the literature as Louvain method or BGLL. This method has two phases. The first phase tries to move all nodes from their community to the one that best improves the modularity maximization objective function. The nodes are selected randomly. If an improvement is found, the first phase is repeated. Otherwise, the second phase starts. In the second phase, all the nodes that belong to the same community are merged, composing a meta-node. The edges between the nodes, which now belong to a meta-node, have their weights accumulated in a single meta-edge. A self-loop-edge is used to represent the internal edges between the nodes of the same community. After that, the meta-nodes are considered nodes for a new execution of the first phase. When no improvement is found, the method stops. The heuristic is known to solve graphs with hundreds of thousands of nodes.

Rotta and Noack [2011] performed one of the most important analyses among the universe of the best-known heuristics for modularity maximization. They classified, combined, and tested greedy and local search heuristics for modularity maximization in different evaluation functions. The modularity value and the time required are used during their arguments when analyzing the results of the experiments performed. They combined greedy and local search methods into a heuristic called multilevel. This heuristic used the merging steps from a greedy search to refine the solutions with the local search methods. Heuristics that have followed this design obtained the highest modularity score.

The modularity maximization problem also received some efforts in the field of exact methods. Xu et al. [2007] suggested a mixed integer quadratic programming model based on the edges to define the variables. After that, Brandes et al. [2008] defined an integer nonlinear programming model, where the number of variables is $|V|^2$, and V is the set of nodes from the graph instance. Aloise et al. [2010] created column generation models that used heuristics to quickly find improved variables to insert into the model that solved the larger instances than previous works. They solved instances with up to 512 nodes.

3. Evolutionary Computation

The desire to employ concepts related to evolution in optimization strategies has been present in the field of Computer Science since the decade of 1960. Around this date, John Holland

involved himself in studies concerning the formalization of adaptive mechanisms usually encountered in nature and how computers might be used to simulate these phenomena. His efforts resulted in the invention of *genetic algorithms*, which became a widely used optimization technique and gave rise to the field of Evolutionary Computation [Goldberg, 1989; Mitchell, 1998].

Various other evolutionary optimization methods were developed since the advent of genetic algorithms. This list includes genetic-inspired techniques such as Evolutionary Programming and Evolution Strategies, swarm-based algorithms like Particle Swarm Optimization and Ant Colony Optimization, and other heuristics such as Differential Evolution and Harmony Search. While each one of these methods has their own intricacies, some characteristics are shared with them.

All EC heuristics are population-based. This means that instead of trying to improve upon a single solution, as in local searches, population-based methods use a number of solutions that aid each other in some way during the process of optimization [Back et al., 1997]. These solutions are the individuals inside of the population. The manner in which individuals cooperate depends on the algorithm; the *crossover operator* is the usual approach, introduced with Holland's genetic algorithms and adopted by various methods. A crossover between two individuals generates one or more offspring arranged in a specific way, containing parts of the original individuals. This operator is based on the belief that a combination between two good solutions has a high probability of resulting in an even better solution [Mitchell, 1998].

The population aspect of EC methods provides a robust *exploration* because it allows to have solutions from multiple regions of the search space. Along with exploration, an effective search procedure must also offer the capability to *exploit* a limited region of the search space, in an attempt to improve an already promising solution. These two characteristics combined are what differ metaheuristics (and by extension, EC methods) from heuristics, enabling them to escape from locally optimal solutions.

In the context of EC, the exploitation component of the search is specific to each strategy. Genetic algorithms and similar methods achieve this capability through a combination of the previously described crossover operator and the *selection operator*, which is analogous to the natural selection process observed in nature. The selection operator is responsible for choosing individuals from the population to undergo reproduction through the crossover operator [Goldberg, 1989]. Even though there is no unique definition of how the selection operator should behave, the search procedure is only effective if it prioritizes individuals with the best fitness values (solutions closer to the global optimum). However, the other individuals should also have a chance to be selected; otherwise, the exploration aspect of the algorithm would be compromised.

An aspect that distinguishes genetic algorithms and other EC strategies from traditional search methods is the improvement of candidate solutions. It does not depend on auxiliary information about the problem, such as the neighborhood of a solution, as is the case in local searches like hill climbing [Goldberg, 1989]. In contrast, evolution happens through the combination of characteristics from specific individuals, using the previously explained crossover operator. Thus, instead of moving smoothly through the search space, solutions repeatedly *jump* to different (usually better) positions, since they receive significant changes at each iteration of the process.

Establishing a balance between exploration and exploitation is an essential task in the design of a robust search procedure. Fundamentally, the goal of such an algorithm is to avoid getting stuck in local optima that are far from the global optimum, since such a condition would cause the optimization process to prematurely terminate with a solution that is usually far from satisfactory [Goldberg, 1989]. Thus, an algorithm that possesses an extremely performant explorative capacity will almost always be able to escape from local optima, but doing so would render it incapable of improving a specific solution, causing it to behave like a random search. In contrast, an algorithm that does not efficiently explore the search space of the problem would become stuck in the first encountered local optimum, no matter how powerful its exploitation ability is.

In complex problems, whose search spaces generally have multiple dimensions and numerous local optima, EC algorithms typically outperform simpler search procedures that have no explorative ability [Goldberg, 1989]. The community detection problem is usually regarded as extremely complex, given the elevated number of publications that attempt to solve it using a wide range of techniques. As such, it would be reasonable to suppose that EC algorithms would be able to solve it effectively. This assumption will be explored in the following sections.

4. Detecting Communities with Evolutionary Computation Methods

Given the expressive number of papers that suggest the usage of EC strategies to detect communities through modularity maximization and their evident effectiveness in solving difficult optimization problems, one might assume that they have an excellent performance in the context of modularity maximization. However, surveying the literature, one might also notice that the proposed algorithms frequently have to be adapted in some manner to be able to adequately maximize modularity. The improvement in quality they provide is often remarkably small when compared to competing methods (usually non-evolutionary), while also exhibiting longer running times. In this section, we introduce a few of these papers, and also point out some characteristics of the modularity function that may contribute to the aforementioned issues.

One of the first attempts to employ EC algorithms for solving the modularity maximization problem was made by Pizzuti [2008]. The author proposed a genetic algorithm named GA-Net which used a metric called *community score* as the fitness (quality) function, also introduced by the author. Along with synthetic networks, three real-world networks were used in the experiments; however the author did not publish the run-time of the algorithm, and the only quality measure presented was Normalized Mutual Information (NMI). In a later work by Zhou and Wang [2016], the authors introduced an algorithm based on Particle Swarm Optimization, comparing it with GA-Net and other algorithms, providing running times and modularity values for both methods. In the next section, both studies will be explored.

Another study involving the usage of EC algorithms was carried out by Gach and Hao [2012], using a hybrid genetic algorithm. Their approach combines the crossover operator with a local search procedure, improving the intensification aspect of the heuristic. The authors employed the Louvain method [Blondel et al., 2008] as the local search and also as the initial solution generator. They reported experimental results with 11 real-world networks with up to 27519 nodes, presenting both the maximum modularity encountered and the running time of their algorithm. However, compared to the Louvain method by itself, their algorithm improved the modularity value by less than 1%. We comment this result further in the next section, relating it to the properties of the search space of the problem.

In Atay et al. [2017], six different EC algorithms were developed to solve the modularity maximization problem. The authors argued that due to the $\mathcal{NP}$-hardness of the problem, the usage of these algorithms is an appropriate approach for solving it. They also stated that four of the six algorithms had to be modified, either by employing new techniques or by including other strategies, to be able to properly find optimal solutions. Their experiments involved nine networks with up to 732 nodes, taking an average of 4 hours to find solutions for a network of this size.

Local searches have obtained suboptimal solutions for modularity maximization in the literature [Duch and Arenas, 2005; Reichardt and Bornholdt, 2006; Blondel et al., 2008; Santiago and Lamb, 2016a]. These results suggest that they are able to access the region of suboptimal solutions easily. This region has an exponential suboptimal number of solutions [Good et al., 2010; Santiago and Lamb, 2016b]. Reaching this area can be done by changing the highest degree ranked nodes [Santiago and Lamb, 2016b], so a local search can move these nodes from low-scored clusters to better ones. Usually, complex networks have a greater number of small degree nodes than high degree nodes.

Fortunato and Hric [2016] also provide a discussion on the topic of the landscape of the modularity function. Continuing the explorative analysis initiated by Good et al. [2010], the authors

comment that identifying the globally optimal solution for the modularity maximization problem is nearly impossible, but finding high-quality partitions in terms of modularity is relatively easy. The authors also state that partitions with similar modularity values may be completely distinct from each other. This implies that none of these nearly optimal partitions can be expected to represent the correct community structure of a network; thus, a marginal increase in the modularity value of a partition is practically meaningless.

Taking these observations about modularity into account, we investigate several publications related to the usage of EC strategies in the context of this problem, discussing their results in terms of consistency and relevance. Furthermore, we perform experiments with a basic local search, aiming to identify the difficulty of finding a high-quality partition of a network, as well as establishing how likely it is for a search with no explorative ability to become stuck in local optima, using eight real-world networks as problem instances. Our results are shown in the next section.

5. Analysis and Experimental Evaluation

In this section, we critically evaluate selected papers that propose the utilization of EC algorithms to solve the community detection problem through modularity maximization. We also provide comments on the topology of the search space and comparisons to alternative algorithms, regarding quality and speed.

5.1. Preliminaries

Papers proposing algorithms to solve the community detection problem frequently employ computational experiments to demonstrate the effectiveness of their methods. To facilitate comparisons to alternative techniques, researchers have come to a general consensus about a few real-world networks to be used as problem instances in the experiments. Table 1 presents some of the most frequently observed networks in the literature, along with their best-known modularity value (Q), the number of nodes (n) and the number connections (m). Network names might slightly differ.

Table 1: Real-world networks commonly used in the literature

Network	**Q**	***n***	***m***
Zachary's Karate Club	0.4198*	34	78
Bottlenose Dolphins	0.5286*	62	159
Politics Books	0.5273*	105	441
American College Football	0.6046*	115	613
C. Elegans Metabolic	0.4531	453	2025
E-mail Network	0.5828	1133	5451

An asterisk in the modularity value indicates that the value was obtained by an exact method [Aloise et al., 2010], and thus it cannot be exceeded. The other three values were obtained through the memetic algorithm proposed by Gach and Hao [2012], also explored in this paper.

In order to attain a better comprehension of the search space topology of the modularity function, we developed and ran experiments with a simple local search (SLS) with no explorative ability, and thus unable to escape local optima. The design details of this heuristic are presented in the following subsection. We performed 100 experiments for each network in Table 1 on a computer with an Intel Core 2 Duo E8400 3.00 GHz processor, running Ubuntu 16.04 LTS. On the same computer, we also performed experiments with CNM and Louvain heuristics. The latter heuristic was tested 30 times due to its probabilistic property.

Table 2 shows the average results obtained by the tested heuristics (SLS, CNM, and Louvain). These results are also compared results of other heuristics from the literature. The table is divided into three parts. The first part shows the optimal results obtained by Aloise et al. [2010] and the best-known results from the literature. The second part shows results of constructive and

greedy local search heuristics (GRASP+PR and VNDS), including the tested ones (SLS, CNM, and Louvain). The third part shows the results of EC algorithms found in the literature (GSA, BA, BB-BC, SSGA, BADE, HDSA, DPSO, GA, PSO). The gaps are calculated by using the referred value divided by the best-known Q value, multiplied by 100. A dash (–) indicates that the authors did not use the network in question in their experiments. The symbol $\sim$ is used to indicate that the values are obtained by observing the results from charts in the selected paper.

Table 2: Comparison between non-evolutionary heuristics and EC algorithms. The reference for each method is idenfified as following: 1. Clauset et al. [2004]; 2. Blondel et al. [2008]; 3. Nascimento and Pitsoulis [2013]; 4. Aloise et al. [2013]; 5. Atay et al. [2017]; 6. Gach and Hao [2012]; 7. Zhou and Wang [2016]; 8. Pizzuti [2008].

	Karate			Dolphins			Polbooks		
	Q	**%gap**	**T(s)**	**Q**	**%gap**	**T(s)**	**Q**	**%gap**	**T(s)**
Optimal	**.4198**		.34	.5285		7.75	.5272		45.65
Best-known	**.4198**			**.5286**			**.5273**		
SLS	.3824±0.0022	8.89%	**.00**	.4988±0.0021	5.62%	.005	.5063±0.0013	3.98%	.02
CNM[1]	.3806	9.32%	.00	.4954	6.26%	.00	.5019	4.80%	.001
Louvain[2]	.4155±0.0	1.00%	.002	.5233±0.0	1.00%	**.0001**	.5266±0.0	0.13%	**.00033**
GRASP+PR[3]	**.4198**	0.00%	.00	.5285	0.02%	.00	.5270	0.06%	1.00
VNDS[4]	.4197	0.00%	.00	.5285	0.02%	.00	.5272	0.01%	.00
GSA[5]	.4170	0.67%	39.88	.4677	11.52%	104.17	.4661	11.61%	309.82
BA[5]	.4133	1.55%	37.22	.4917	6.98%	105.72	.5020	4.80%	279.34
BB-BC[5]	.4196	0.05%	46.22	.5141	2.74%	128.68	.4914	6.81%	373.51
SSGA[5]	**.4198**	0.00%	57.40	.5200	1.63%	139.96	.5203	1.33%	469.04
BADE[5]	.4188	0.24%	48.00	.5128	2.99%	135.86	.5178	1.80%	372.28
HDSA[5]	**.4198**	0.00%	45.20	.5282	0.08%	128.16	.5272	0.02%	371.55
MA-COM[6]	**.4198**	0.00%	.30	**.5286**	0.00%	.50	**.5273**	0.00%	1.40
DPSO[7]	.4132	1.57%	$\sim$ 10	.5109	3.35%	$\sim$ 20	.5235	0.72%	$\sim$ 40
GA[8,7]	.4060	3.29%	$\sim$ 40	.4634	12.33%	$\sim$ 105	.4798	9.01%	$\sim$ 150
PSO[7]	.3835	8.65%	$\sim$ 10	.5114	3.25%	$\sim$ 20	.4969	5.77%	$\sim$ 40

	Football			Celegans Metabolic			Email		
	Q	**%gap**	**T(s)**	**Q**	**%gap**	**T(s)**	**Q**	**%gap**	**T(s)**
Optimal	**.6046**		249.41	-	-	-	-	-	-
Best-known	**.6046**			**.4531**			**.5828**		
SLS	.5731±0.0017	5.20%	.02	.3952±0.001	12.77%	.73	.5480±0.0008	5.97%	11.55
CNM[1]	.5772	4.52%	**.001**	.4058	10.44%	.009	.5147	11.67%	.04
Louvain[2]	.6044±0.0	0.03%	**.001**	.4350±0.0	3.99%	**.0045**	.5695±0.00004	2.28%	**.01**
GRASP+PR[3]	**.6046**	0.00%	1.00	.4520	0.24%	98.00	.5820	0.14%	1565.00
VNDS[4]	.6045	0.00%	.00	.4528	0.04%	1.97	.4271	26.71%	1.69
GSA[5]	.4032	33.31%	337.89	.3039	32.93%	5297.44	–	–	–
BA[5]	.5272	12.80%	333.43	.3438	24.12%	6413.49	–	–	–
BB-BC[5]	.5061	16.29%	450.71	.3266	27.92%	8245.63	–	–	–
SSGA[5]	.5277	12.72%	521.48	.3220	28.93%	10074.91	–	–	–
BADE[5]	.5513	8.82%	433.54	.3385	25.29%	9265.17	–	–	–
HDSA[5]	.6033	0.22%	437.76	.4074	10.09%	8256.21	–	–	–
MA-COM[6]	**.6046**	0.00%	1.00	**.4531**	0.00%	8.30	**.5828**	0.00%	23.10
DPSO[7]	.6015	0.51%	$\sim$ 55	–	–	–	–	–	–
GA[8,7]	.5906	2.32%	$\sim$ 165	–	–	–	–	–	–
PSO[7]	.6021	0.41%	$\sim$ 60	–	–	–	–	–	–

5.2. Simple Local Search

In Algorithm 1, the local search used in this paper is shown. It uses an one-neighborhood strategy, lacking any ability to escape local optima, in which the search is done by moving nodes to other clusters. The best movement is selected by using the highest value obtained by the ΔQ function, defined by Equation 2. The candidate movements are stored in a fibonacci heap ($heap$). At the start of this local search, the starting solution (si) is considered the best solution (s_{best}). Every candidate movement is inserted into a fibonacci heap (by the $startFibonacciHeap$ function). Then an iterative phase starts, where the best movement is selected from the $heap$. If this movement improves the objective function, then a new iteration is executed; otherwise, the procedure stops.

Before a new iteration is done, the new movement options are inserted into the fibonacci heap (*updateHeap*).

Algorithm 1: Simple Local Search

Input : $G(V, E)$, si
$s_{best} \leftarrow si$
$Q_{s_{best}} \leftarrow Q(si)$
$heap \leftarrow startFibonacciHeap(si)$
while *not* $heap.empty()$ **do**
  $\{s, \Delta Q\} \leftarrow heap.pop()$
  $Q_s \leftarrow Q_{s_{best}} + \Delta Q$
  if $Q_s \leq Q_{s_{best}}$ **then**
    break
  $s_{best} \leftarrow s$
  $Q_{s_{best}} \leftarrow Q_s$
  $updateHeap(s_{best})$
return s_{best}

To avoid $\Theta(n^2)$ operations, the function ΔQ is defined to quickly calculate the best neighbor community to move node v from community C_v to C_k, where d_w is the degree of node $w \in V$.

$$\Delta Q(v,k) = \frac{1}{|E|}\left[\sum_{u \in C_k}\left(a_{vu} - \frac{d_v d_u}{2|E|}\right) - \sum_{u \in C_v}\left(a_{vu} - \frac{d_v d_u}{2|E|}\right)\right] \tag{2}$$

5.3. Analysis on Evolutionary Computation Methods

The first paper analyzed is about one of the earliest attempts at developing a genetic algorithm for the community detection problem called GA-Net, proposed by Pizzuti [2008]. The author described a standard genetic algorithm with an additional characteristic: if two individuals i and j belong to the same community, they must also have a connection between them. Thus, the operators of crossover and mutation had to be adapted to take this constraint into account. The GA-Net algorithm starts with a randomly generated population. Instead of modularity, a new evaluation metric named community score was used as the fitness function, and the author did not present the run-time of the algorithm. However, both time and modularity values could be found in a paper by Zhou and Wang [2016]; our comments are based on these results. The real-world networks tested were Zachary's Karate Club, Bottlenose Dolphins, Politics Books and American College Football. The exact modularity values are known for these networks, shown in Table 1. GA-Net was able to approach these values by a small margin of approximately 2% to 12%, indicating that it is capable of finding solutions close to the global optimum, but unable to identify it exactly. In the smallest network (Karate Club), Zhou and Wang [2016] reports a run-time of approximately 50 seconds, while in the largest one (College Football), the algorithm took approximately 165 seconds to solve the problem.

While the results suggest that the GA-Net algorithm is good at finding relatively high-quality solutions, it takes an extremely long time do so, even in networks with less than 106 nodes. Compared to our local search, the difference in quality is less than 5% on average, but its run-time is several orders of magnitude higher, as the simple local search takes much less than one second in every case. This observation raises multiple questions about the effectiveness of genetic algorithms in the modularity maximization problem. For instance, we might wonder why a simple local search with absolutely no ability to escape from local minima was able to find solutions with a modularity value so close to the ones obtained through a genetic algorithm, which is a much more robust search

procedure. We might also question whether or not an improvement of 5% is worth of a running time that is more than a thousand times slower, considering the degeneracies of the modularity function.

Gach and Hao [2012] proposed a memetic algorithm, named MA-COM, which combines a genetic algorithm with the Louvain method. The genetic algorithm portion of their proposed heuristic is significantly different from a typical implementation: the selection operator is completely random, making no distinction between good and bad solutions; there is no mutation operator, slightly limiting its ability to explore the search space; and the crossover operator is meant to generate a solution *worse* than its parents, since it tries to increase the number of communities.

The Louvain method is used both to generate the initial population and to improve solutions generated through the crossover operator. This means that the MA-COM algorithm already starts with a set of extremely high-quality solutions, as multiple papers have shown that the Louvain method is one of the most effective algorithms for detecting communities, both in quality and speed [Fortunato and Hric, 2016; Yang et al., 2016]. Running it after every crossover operation also guarantees that solutions will always have a high modularity value, thereby rectifying the deterioration in quality caused by the higher number of communities. The results presented by the authors indicate that the memetic algorithm provides absolutely no improvement over the Louvain method in networks with less than 200 nodes. In the remaining six networks, an average improvement of approximately 1% was observed. Taking into account the characteristics of the search space of the modularity function described throughout this and related papers, such as the fact that the region of partitions with nearly optimal modularity values is more than exponentially large, we can infer that the partitions obtained through the algorithm are highly likely to be still incorrect, since they are not guaranteed to be globally optimal.

In Zhou and Wang [2016] a new method based on discrete PSO is proposed, named as DPSO here. The authors used a method similar to label propagation to generate initial solutions, pointing out that the use of good early solutions can reduce the searching time significantly. The strategy used to initialize and update the swarm is based on the impact of each node, where all nodes receive a tag as a reference. Each node is sorted by its rank in descending order. The node with the highest degree assigns its tag to the nodes that have a higher degree of connection. However, the authors do not know when the algorithm converges to a stable state on different networks, and due to this reason, they used a different update strategy, proposed by Gong et al. [2012], in uneven iterations.

Also in Zhou and Wang [2016], the authors stated that modularity density (D) was used as the objective function for the algorithm, but they employ Q and NMI when presenting their results. Compared to the proposed algorithm, our simple local search was able to find solutions with slightly less quality, with an average difference of approximately 5%. Moreover, when comparing their results to other heuristics, the authors present modularity values higher than the exact values, identified by Aloise et al. [2010], for some combinations of heuristic and network, especially in the College Football instance. This is a serious inconsistency that may raise questions about the legitimacy of their experiments.

Atay et al. [2017] proposed six EC heuristics to solve the modularity maximization problem. They are Bat Algorithm (BA), Gravitational Search Algorithm (GSA), Big Bang-Big Crunch (BB-BC), Bat Algorithm based on Differential Evolutionary Algorithm (BADE), Hyperheuristic Differential Search Algorithm (HDSA), and Scatter Search based on Genetic Algorithm (SSGA). The authors do not compare their results with other methods in the literature, so the claim about showing "the success of effective hyperheuristic differential search algorithm which has offered the solutions containing the best modularity" cannot be supported. For some classical networks, the run-time required to obtain solutions at most 5% better than the Louvain method was extraordinarily higher than the reported in the modularity maximization literature (see Table 2). Considering the existence of an exponential number of suboptimal solutions, we cannot rely on the notion that small improvements in modularity score mean better solutions than other suboptimal ones.

6. Discussion and Concluding Remarks

In this paper, we explored the characteristics of the modularity function and how effective Evolutionary Computation (EC) algorithms are. We critically reviewed papers which proposed the usage of these algorithms and observed either insignificant quality improvements or high run-times when compared to non-evolutionary algorithms like the Louvain method [Blondel et al., 2008].

The most important aspect of EC strategies is that they are extremely effective at exploring the search space of the problem, easily escaping from local optima while still having a fairly satisfactory run-time. However, experimenting a local search without any explorative ability in networks with up to 1133 nodes, we observed an absence of local optima in the region of solutions with up to approximately 90% of the modularity value of the best-known partition. This result can be clearly seen in Table 2. The space above this region is likely to be the plateau comprised of an exponential number of solutions, initially described by Good et al. [2010].

Since this region of high-quality partitions can be easily accessed through greedy or constructive local search heuristics like CNM, the Louvain method and even by a simple local search (see Sections 2 and 5.2), the usage of EC algorithms may be questionable. Most EC strategies demand an excessive amount of computational time to find acceptable solutions, unless they incorporate non-evolutionary techniques, and even so the improvement they provide (if any) is too small to justify the time and effort required.

The modularity function itself is also a deeply controversial topic, since it displays some degeneracies that limit its ability to identify the communities of a network correctly. Nevertheless, detecting communities by maximizing modularity may be feasible in applications where time is a priority and approximate solutions are acceptable, such as in directed advertising or in the division of interconnected systems (e.g. metabolic networks and electrical grids) into smaller modules with the intent of facilitating their study.

As further works, we suggest the study of the behavior of several heuristics and metaheuristics in the search space of the modularity maximization problem to understand which patterns are followed by them. It can be used to improve the search, and to identify how and when to use greedy, local search, or metaheuristics like EC strategies.

References

Achacoso, T. B. and Yamamoto, W. S. (1991). *AY's Neuroanatomy of C. elegans for Computation*. CRC Press.

Agarwal, G. and Kempe, D. (2008). Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, 66(3):409–418. ISSN 1434-6028.

Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., and Liberti, L. (2010). Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4): 046112. ISSN 1539-3755.

Aloise, D., Caporossi, G., Hansen, P., Liberti, L., Perron, S., and Ruiz, M. (2013). Modularity maximization in networks by variable neighborhood search. In Bader, D. A., Meyerhenke, H., Sanders, P., and Wagner, D., editors, *Contemporary Mathematics*, volume 588, chapter Graph Partitioning and Graph Clustering, p. 113–127.

Atay, Y., Koc, I., Babaoglu, I., and Kodaz, H. (2017). Community detection from biological and social networks. *Appl. Soft Comput.*, 50(C):194–211. ISSN 1568-4946.

Back, T., Fogel, D. B., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition. ISBN 0750303921.

Bedi, P. and Sharma, C. (2016). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(3):115–135. ISSN 1942-4795.

Bell, D. C., Atkinson, J. S., and Carlson, J. W. (1999). Centrality measures for disease transmission networks. *Social Networks*, 21(1):1–21. ISSN 03788733.

Bishop, A. N. and Shames, I. (2011). Link operations for slowing the spread of disease in complex networks. *EPL (Europhysics Letters)*, 95(1):18005.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008. ISSN 1742-5468.

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. (2008). On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188. ISSN 1041-4347.

Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6):066111–1 —- 066111–6. ISSN 1539-3755.

Djidjev, H. N. and Onus, M. (2013). Scalable and Accurate Graph Clustering and Community Structure Detection. *IEEE Transactions on Parallel and Distributed Systems*, 24(5):1022–1029. ISSN 1045-9219.

Duch, J. and Arenas, A. (2005). Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2):2–5. ISSN 1539-3755.

Ferrara, E., De Meo, P., Catanese, S., and Fiumara, G. (2014). Detecting criminal organizations in mobile phone networks. *Expert Systems with Applications*, 41(13):5733–5750. ISSN 09574174.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*.

Fortunato, S. and Barthélemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America*, 104(1):36–41. ISSN 0027-8424.

Fortunato, S. and Hric, D. (2016). Community detection in networks: A user guide. *Physics Reports*, 659:1 – 44. ISSN 0370-1573. Community detection in networks: A user guide.

Gach, O. and Hao, J.-K. (2012). *A Memetic Algorithm for Community Detection in Complex Networks*, p. 327–336. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-32964-7.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition. ISBN 0201157675.

Gong, M., Cai, Q., Li, Y., and Ma, J. (2012). An improved memetic algorithm for community detection in complex networks. In *2012 IEEE Congress on Evolutionary Computation*, p. 1–8, Brisbane, QLD. IEEE. ISBN 978-1-4673-1509-8.

Good, B. H., de Montjoye, Y.-A., and Clauset, A. (2010). Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106. ISSN 1539-3755.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabasi, A.-L. (2000). The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654. ISSN 0028-0836.

Marcelino, J. and Kaiser, M. (2012). Critical paths in a metapopulation model of H1N1: Efficiently delaying influenza spreading through flight cancellation. *PLoS Currents*. ISSN 21573999.

Meunier, D., Fonlupt, P., Saive, A.-L., Plailly, J., Ravel, N., and Royet, J.-P. (2014). Modular structure of functional networks in olfactory memory. *NeuroImage*, 95:264–75. ISSN 1095-9572.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA. ISBN 0262631857.

Nascimento, M. C. and Pitsoulis, L. (2013). Community detection by modularity maximization using GRASP with path relinking. *Computers & Operations Research*, 40(12):3121–3131. ISSN 03050548.

Nepusz, T., Yu, H., and Paccanaro, A. (2012). Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods 9*, p. 471–472.

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113.

Newman, M. E. (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409.

Newman, M. (2013). Spectral community detection in sparse networks. *arXiv preprint arXiv:1308.6494*.

Pastor-Satorras, R., Castellano, C., Van Mieghem, P., and Vespignani, A. (2015). Epidemic processes in complex networks. *Reviews of Modern Physics*, 87(3):925–979. ISSN 0034-6861.

Pizzuti, C. (2008). Ga-net: A genetic algorithm for community detection in social networks. In *International Conference on Parallel Problem Solving from Nature*, p. 1081–1090. Springer.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663. ISSN 0027-8424.

Reichardt, J. and Bornholdt, S. (2006). Statistical Mechanics of Community Detection. p. 1–16.

Rotta, R. and Noack, A. (2011). Multilevel local search algorithms for modularity clustering. *Journal of Experimental Algorithmics*, 16(2):2.1. ISSN 10846654.

Santiago, R. and Lamb, L. C. (2016a). Efficient Stochastic Local Search for Modularity Maximization. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion - GECCO '16 Companion*, p. 51–52, New York, New York, USA. ACM Press. ISBN 9781450343237.

Santiago, R. D. and Lamb, L. C. (2016b). On the Role of Degree Influence in Suboptimal Modularity Maximization. In *IEEE Congress on Evolutionary Computation*, p. 4618–4625, Vancouver. IEEE. ISBN 978-1-5090-0622-9.

Xu, G., Tsoka, S., and Papageorgiou, L. G. (2007). Finding community structures in complex networks using mixed integer optimisation. *The European Physical Journal B*, 60(2):231–239. ISSN 1434-6036.

Yang, Z., Algesheimer, R., and Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6. Article.

Zhou, D. and Wang, X. (2016). A neighborhood-impact based community detection algorithm via discrete PSO. *Mathematical Problems in Engineering*, 2016:1–15.