Analyzing genetic and single solution algorithms in Community Detection Problems

Lukas Arana
larana013@ikasle.ehu.eus
Euskal Herriko Unibertsitatea
Donostia, Gipuzkoa, Euskal Herria

Oier Ijurko
oijurco001@ikasle.ehu.eus
Euskal Herriko Unibertsitatea
Donostia, Gipuzkoa, Euskal Herria

ABSTRACT

Community detection techniques are useful for social media algorithms to discover people with common interests and keep them tightly connected. Due to the low popularity of Community Detection problems, this type of problems have been limited to few works about them. In this paper, we present the idea that Unique Solution Algorithms can be more efficient than Population Based Algorithms when working with Community Detection problems.
• ;

1 INTRODUCTION

Community detection is a way to analyze different networks by examining the relationship of each individual who forms the network with the other individuals in the network. Assigning each individual to a group is the main goal of the problem, maximizing the internal relationships of each group while minimizing external relations. Finding an effective algorithm to solve this kind of problems could be crucial in many aspects of modern life.

2 FORMALIZATION

This problem has many different ways to be mathematically formalized, graph theory being the most widely used of them all. A network is represented as a graph G(v, e), where v is the node group and e the vertices group. We will represent each node as an individual in the network and will indicate all the relationships between individuals.

By doing this, we can define a community as a group of graph nodes. Therefore, the problem of detecting communities consists of maximizing the relationship between members of the same community and minimizing the relationship with external members.

Due to efficiency issues we have also used a set list of sets as the codification of our problem, where each set keeps the names of the authors of a community. As mentioned earlier, the reason for this codification is its effectiveness; it is much more practical to work with the NetworkX library, which is very useful when working with graphs.

It should be noted that all the solutions we obtain will be viable to our problem, that is, we will not have to deal explicitly with the viability of the solutions. Moreover, the solution space of our problem will be p^n, with p being the number of partitions in the network and n being the number of authors.

, ,
© 2022
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Redundancy can also be detected between a few solutions. Partitions can be made with the same authors on behalf of a different partition. For example, partitions [0,1,0] and [1,0,1] represent the same, changing only their values . As this is rare case, we have decided not to treat it.

Community detection, although new, has been the subject of study for decades, so different methods for evaluating solutions have been developed. The objective function we have chosen has been Modularity. Modularity is a measure of the structure of networks or graphs calculated by dividing networks into modes and measuring their strength. The strength in this case is measured by the number of connections between nodes and their weights. The weights are defined by the number of collaborations between authors.

Modularity is both a property of a network, and a way to divide a network into communities. As mentioned earlier, it measures the quality of a division, depending on whether there are many or few edges within communities. High modularity values correspond to a good fragmentation of the network community and we will use these values to determine if a candidate solution improves the current one. It seems difficult to perform an adequate evaluation of all possible divisions, but modularity allows us to perform this evaluation in an appropriate way and not very costly computationally. The values obtained will be between -0.5 and 1, with the aim of maximizing this value. Non-zero values represent deviations from randomness and, in practice, a modularity value of above 0.3 is indicative of a good community structure.

The equation of modularity (1) and the simplified modularity ultimately calculated (2) are:

$$(1) Q = \frac{1}{2m} \sum_{vw} [A_{vw} * \frac{k_v k_w}{2m}] \delta(c_v c_w)$$
$$(2) = \sum_i (e_{ii} - a_i^2)$$

The value of A_{vw} will be 1 if v and w are connected, and 0 otherwise. The degree k_v of a vertex v is defined as the number of edges incident upon it. Another parameter is m, which is the number of edges in the graph. The δ function will take two vertices and see if they are both from the same community, returning a value of 1 if so, and 0 otherwise.

Although this problem is known and has been subject of research by many, it is still an unsolved problem. No effective algorithms have yet been found to solve this problem and it has been demonstrated that the most studied algorithms currently are NP-HARD.

The algorithms that we are going to experiment with will be EDA and Simulated Annealing.

EDA belongs to the class of evolutionary algorithms. The main difference between EDAs and other evolutionary algorithms is that the way to generate new candidate solutions uses an explicit probability distribution encoded by a Bayesian network, a multivariate normal distribution, or another model class. For this algorithm we will create an initial population, evaluate the candidates, get the best candidates in the population and create a new population using a probabilistic distribution based on the best candidates in the current population.

Simulated Annealing is a metaheuristic to approximate global optimization in a large search space for an optimization problem. The name of the algorithm comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to alter its physical properties. Both are attributes of the material that depend on their thermodynamic free energy. Heating and cooling the material affects both the temperature and the thermodynamic free energy. This notion of slow cooling implemented in the simulated annealing algorithm is interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored. Accepting worse solutions allows for a more extensive search for the global optimal solution, thus benefiting exploration.

The function used to define a candidate for new solutions takes into account the number of connections between authors and tries to get the authors with most connections between themselves in the same community.

3 EXPERIMENTAL STUDY

3.1 Instance generators

In order to benchmark properly these two algorithms we need to first create adequate testing data. For that purpose random instance generators have been used on top of the given graph. For that purpose, given the desired size of the total population (n), the generator will randomly delete V - n nodes from the graph, leaving the graph with only the desired number of nodes. In this way, we are able to test the algorithms in different instances of desired sizes and therefore analyze not only the generalizability but also the scalability of both algorithms in different situations.

3.2 Experimental Setting

This does not explain the setting used for parameter tunning. That can be found in the corresponding chapter (3.3).

The experiments have been designed with the purpose of testing different aspects of the algorithms while maintaining a correct statistical methodology. As explained above, generators have been used to get a variety of environments while being limited to a single dataset.

We ran all the experiments in an Ubuntu system with an intel i5-1135G7 processor and 16 GB ram memory. All the tests, unless otherwise indicated, show the results obtained by running the corresponding algorithm n = 15 times with a different instance for each run. When comparing the algorithms, both were run with the same 15 instances. This way we get to test the behavior of each algorithm on different environments while only using a single dataset.

For the goal of analyzing scalability, algorithms are run n=15 times with the instance conditions described above with a limiting maximum number of evaluations in the cost function. These maximum evaluations were 1000, 5000, 10000, 20000, 30000 and 40000 and they were chosen in order to make both algorithms able to converge to their optimal solution while taking into account our hardware limitations. As seen in the experiments, this number was found to be near 20000.

3.3 Parameter Tuning

Since EDA and simulated annealing algorithms have two parameters to be defined by the programmer, parameter tuning is required with the purpose of improving efficiency. Tuning both algorithms will also lead to a fairer comparison between them, since the random factor of human choice is eliminated.

Both algorithms have two parameters to be tuned and a grid search approach was used. For the purpose of eliminating randomness from the stochastic algorithms and due to hardware limitations, all parameter choices were in n = 5 different instances of p = 2000 size. Finally, The mean of last solutions' modularity was used as the comparison value.

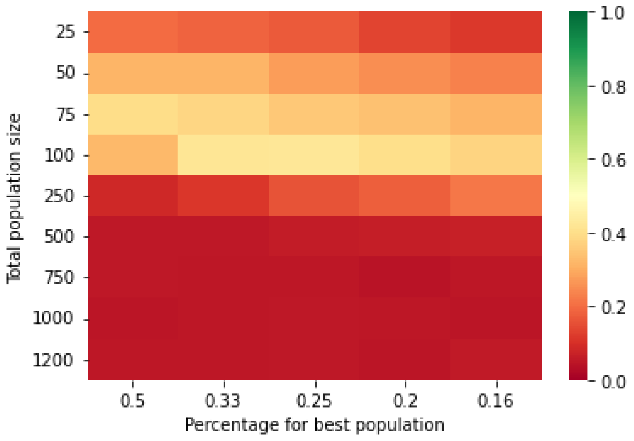


Figure 1
Parameter optimization for EDA algorithm. Different population sizes with percentages of best population

In the EDA algorithm, the parameters to be tuned are the following. 1. Total population size 2. Percentage for best solution. The percentage of the population that will be used as a guideline for the next generation.

As seen in figure 1, experiments show that the optimal population size is 100. When analyzing the percentage used for the next generation values, the values of 0.33 and 0.25 were found to give optimal solutions. In this work, the value of 0.25 has been chosen since the value given by 0.33 was slightly lower.

These results, a low population size and low percentage, indicate that the algorithm does not benefit from high exploration rates when executed for a maximum of 5000 iterations. It is possible that a higher maximum iteration value could give the algorithm more time to explore, leading to a higher genetic diversity and therefore better results when converging.

In the simulated annealing the hyperparameters that can be changed were the following: Beta: This parameter controls the temperature decay rate of the algorithm as explained above. P: The initial probability for a certain answer to be accepted.

While doing the experiments we have seen no significant change between a parameter or another. This is due to the fast decay of temperature shown in figure 2. It shows how the Simulated Annealing algorithm already converges on an optimal solution with 10000 evaluations and therefore we expect no big changes on the optimal solution in further evaluations.

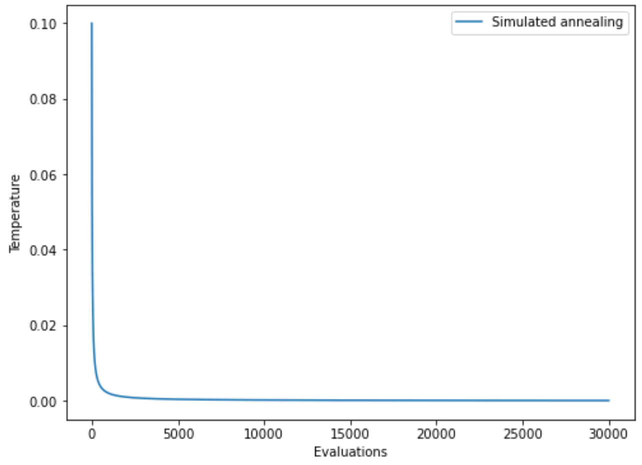


Figure 2
The plot shows how the temperature decays through a run. A single run in a single instance is shown since there is not a significant difference between them.

Therefore, since a low beta is expected to increase the SA's exploration rates, the chosen parameter values are $\beta = 0.1 and P = 0.5$.

3.4 Unique Solution Algorithm effectiveness

Following the experimentation guide described in this chapter, various tests were run with the intention of analyzing and comparing the two algorithms. Indeed, the goal is to compare their effectiveness and extrapolate the results obtained. Effectiveness in an algorithm is defined by two different measures that we will examine independently: the modularity values obtained by an algorithm and the execution time needed to get this solution. Finally, the most effective algorithm will then be the one maximizing the modularity while minimizing the execution time.

In figure 2 we see the modularity values obtained by the final candidate groups in both SA and EDA algorithms; simulated annealing has a faster convergence (only 10000 evaluations are needed) to its optimal solution while EDA needs more evaluations in order to improve its solutions. In fact, we only see similar solutions with 30000 evaluations, where the best 25% solutions from EDA and the worst 25% solutions from Simulated annealing are found to be in the same range. Because of that, we conclude that, in our dataset, the SA dominates the EDA algorithm in terms of modularity values when maximum evaluations are less than 30000.

However, even though the graph shows a decay on the improvement rate of the EDA algorithm in the higher maximum evaluations, we see no clear convergence in these modularity values. This indicates a possible improvement when running the algorithm with

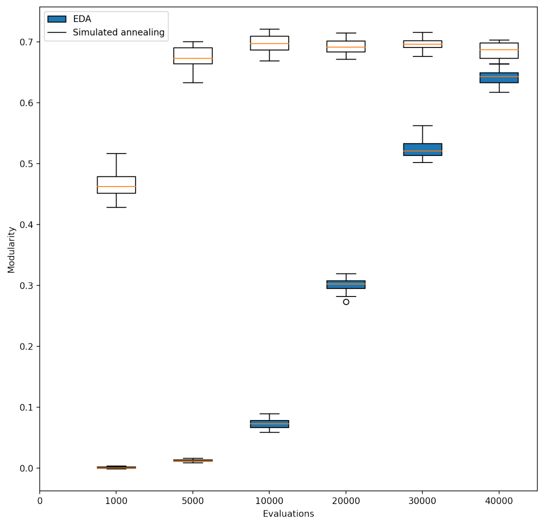


Figure 3
Plot of n=15 solution candidates with different evaluations. X axis represents the evaluations that each candidate group had done.

higher maximum evaluations. Also, the X axis is not linear, meaning that deducing improvement ratio can be misleading.

To gain more insight on the behaviour of these algorithms, we will study the changes on the distribution of the candidate group through different evaluations. To do that, in figure 3 we show the histogram of the solutions in different evaluations in the EDA algorithm. Each histogram represents the modularity values of all the members of current populations in 15 runs with different instances.

According to figure 3, looking at the mean changes in the distributions we can see a linear improvement of its solutions from 5000 to 30000 iterations. However, this graph shows in a more clear way the decay of the improvement ratio after 30000 iterations.

In order to check if this improvement decay means in fact a convergence, we ran new tests in the EDA algorithm with higher maximum evaluations. Especifically, we ran these tests in n = 5 instances and with a maximum of 600000 evaluations.

The figure above shows the mean of the last solutions candidates got by the EDA algorithm when running for 60000 maximum evaluations. The graph shows that the EDA's improvement decays after 40000 evaluations and does worse in these instances than the simulated annealing algorithm in all evaluations. Even though our number of instances is low, given that each instance represents 0.22 of the entire dataset, we can conclude that the EDA algorithm converges after 40000 evaluations.

Regarding the simulated annealing algorithm, very little change is found in their candidates after convergence. As table 4 illustrates, variance values on the candidates differ very little after 5000

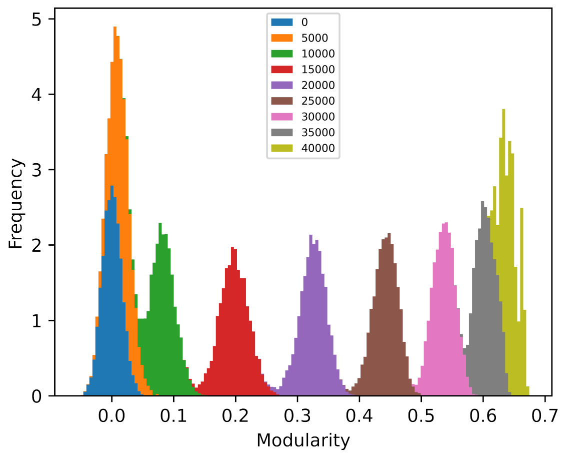
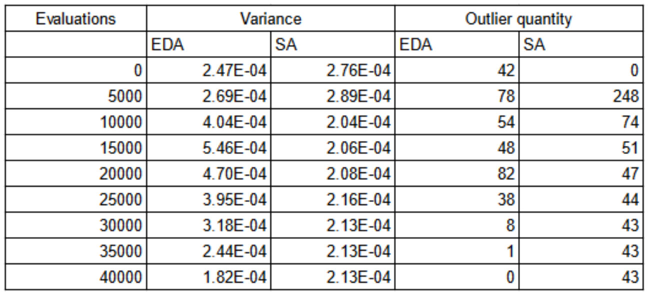


Figure 4
Histograms of solution candidates with different evaluations.

evaluations, as does the amount of outliers. Outliers do not differ significantly after 5000 evaluations. The stability of these two values can be explained by figure 2. That is because as the temperature decays to 0, the candidate solutions group will be less prone to change in the next evaluation. That is why after the temperature gets to its minimum, we see no change at all in the candidate solutions in further evaluations, meaning that the optimal solution for the algorithm is found.



Evaluations	Variance		Outlier quantity	
	EDA	SA	EDA	SA
0	2.47E-04	2.76E-04	42	0
5000	2.69E-04	2.89E-04	78	248
10000	4.04E-04	2.04E-04	54	74
15000	5.46E-04	2.06E-04	48	51
20000	4.70E-04	2.08E-04	82	47
25000	3.95E-04	2.16E-04	38	44
30000	3.18E-04	2.13E-04	8	43
35000	2.44E-04	2.13E-04	1	43
40000	1.82E-04	2.13E-04	0	43

Figure 5
Table showing candidate groups properties for different evaluations. Outliner points are defined as the points with modularity values outside the 25 and 75 quartile of the solution candidate group.

As described in the introduction, the efficacy of an algorithm consists not only of its performance but also of the resources spent to get the solution. In our case, since both algorithms were run on top of the same hardware, we will only take into account the execution time spent by the algorithm.

The execution time spent for each evaluation is shown in figure 5, where each point shows the time spent in each instance. Finally, a line going through the mean of these values is drawn.

In the graph we see that simulated annealing gets consistently lower times for all the evaluations. The difference between both

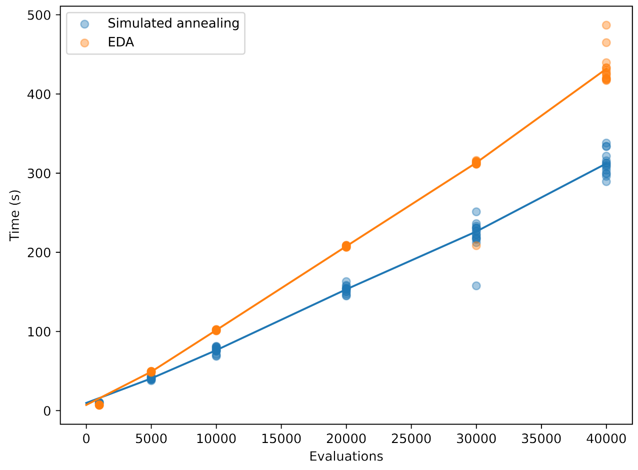


Figure 6
Figure comparing time costs of SA and EDA algorithms.

algorithms' times seems to get bigger as the evaluation amount increases, meaning that the EDA has the worst scalability. It is also worth noting that, since the simulated annealing algorithm converges in around 5000-10000 evaluations, efficiency could significantly be improved by stopping the algorithm when a certain value of temperature is achieved.

4 CONCLUSIONS

This article shows the reasons why in our dataset unique solution algorithms outperform population based algorithms in community detection in both time usage and solution quality. We have shown that unique solution algorithms dominate genetic algorithms in community detection problems even though algorithms such as simulated annealing tend to converge much faster compared to other population based ones in our instances. Our analysis shows that, for our data, community detection can be solved efficiently without high computational resources.

5 REFERENCES

[1] Aaron Clauset, Finding community structure in very large networks, 2004, arXiv:cond-mat/0408187