Journal of Computational Science 31 (2019) 77–85

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Finding influentials in social networks using evolutionary algorithm

Michał Weskida, Radosław Michalski*

Department of Computational Intelligence, Faculty of Computer Science and Management, Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

ARTICLE INFO

Article history:
Received 16 January 2018
Received in revised form
23 November 2018
Accepted 15 December 2018
Available online 18 December 2018

Keywords:
Social networks
Influence maximization
Seed selection
Genetic algorithms
Evolutionary algorithm

ABSTRACT

The social influence maximization problem is an important research topic for many years since it has a tremendous impact on society. As social influence can be maximized for many purposes, such as marketing, politics, spreading innovations, there are many stakeholders interested in progress in this area. As it has been shown, for most settings finding an optimal seed set is an NP-hard problem, this is why various heuristics started to emerge since the statement of the problem. In this work, we explore the applicability of evolutionary algorithm for influence maximization. The experiments conducted using real and artificial social networks and the linear threshold influence model show that this approach offers not only speed and accuracy. Also, some other interesting features have been found, such as the transferability of its parameters to other datasets. Summarizing the results, it was observed that the evolutionary algorithm does not lose its performance when limiting its time by the factor of two and for most datasets, we obtained a high correlation of ranked parameters' sets used for the evolutionary algorithm, typically around 0.9. Overall, these features combined make this approach an interesting research direction for influence maximization.

1. Introduction

As the online social networks are not treated anymore as something separate from our daily life, but rather are now an important part of our life, their role and impact increased significantly over the last decade. From the space in which we seek for escape from daily routines, online social networks became the space that for many of us is the center of activity. The examples include: establishing or maintaining relationships, information or news creation, and retrieval or social activism. At the same time, online social networks are a platform that is extensively used for marketing of any kind, such as for promoting products and services, political campaigns or, more controversially, disinformation. Independently of the purpose and characteristics of those activities, their mechanics is always the same. Namely, given a limited budget for influencing individuals, the outcome of the process should be maximized, and this can relate to a number of (dis)informed individuals, number of voters supporting a given candidate in elections, sales of a promoted product etc. Since this problem, called influence maximization problem, can be formally stated, researchers try to propose the best solutions to tackle it. As it is shown in this work, a significant contribution in terms of efficient heuristics comes from the area of computational intelligence. In this work, we show that the use of genetic algorithms, namely evolutionary algorithm, brings promising results, both in terms of quality and efficiency. These two factors – quality and efficiency – should be considered as equally important nowadays, since online social networks are constantly evolving, this is why the influence maximization algorithms should provide their results within reasonable time constraints. Formerly used assumption about the static nature of networks does not hold anymore and this imposes new challenges for influence maximization algorithms nowadays.

The purpose of this work is to show how the use of evolutionary algorithms improves the quality of results in the influence maximization problem by also providing significant progress in terms of reducing the computational time. Additionally, we demonstrate that the power of evolutionary algorithms lies also in their flexibility understood as the ability to use the algorithm's parameters learned in one network for another network by still providing good results. This shows the transfer learning capabilities of evolutionary algorithms.

In order to achieve the above, this paper raises the question of whether the evolutionary algorithm is a good mean to find a seed set for social influence maximization problem. To answer this question, we conduct a set of experiments using real-world and artificial social networks. For those experiments, we did evaluate

* Corresponding author.
E-mail address: radoslaw.michalski@pwr.edu.pl (R. Michalski).

https://doi.org/10.1016/j.jocs.2018.12.010

nearly 250 evolutionary algorithm model parameters and we have been looking at a number of properties of the approach: (i) how well the evolutionary algorithm performs in the influence maximization problem, (ii) what is the dependency between the quality of results and time given to perform the method, (iii) how transferable are the parameters to other networks.

To sum up the contributions of this paper, these are the following:

- we computationally demonstrate that the evolutionary algorithm is a way to overcome the limitations of the greedy algorithm that builds the seed set iteratively,
- by evaluating the impact of the reduction of time given to evolutionary algorithm on the results, we show that it is able to perform well for situations in which its time for finding seed set is highly reduced, e.g., for real-time decision making,
- the comparison showing how good are the results in the case of transferring the parameters from one network to another, we demonstrate that the power of evolutionary algorithm lies in the possibility of learning the model on one network and running on another if their network models belong to similar families.

This manuscript is organized as follows. Section 2 describes the related work in the area of influence maximization also mentioning early works on the use of genetic algorithms for tackling this problem. Section 3 presents the experimental setup for the experimental part of the work, covering the datasets, influence model settings, evolutionary algorithm's parameters, and the results' evaluation methods. The next section, Section 4, introduces the results and contains the discussion on them, while Section 5 summarizes the article and investigates future work directions.

2. Related work

The influence maximization problem in social networks is studied for over a decade now. As it was already proven [1], in most cases it is an NP-hard problem, so finding its analytic solution is not possible in a polynomial time. Yet, for small networks and static graphs, it is still possible to find the optimal solution by iterating through all the seed sets and computing their spreading performance. Unfortunately, for larger graphs, this approach is computationally too demanding, but there are first attempts to overcome the network size limitations when seeking for exact solution [2].

As it was shown in [1], one of the most intuitive heuristics – computationally demanding *greedy method* – can achieve at least $1 - 1/e - \epsilon$ spread compared to the optimal solution, as the researchers took the advantage of the theory of submodular functions. Since then, the research on influence maximization problem was two-fold. Firstly, researchers tried to optimize the greedy algorithm in order to reduce its running time [3,4], but another line of research focused on finding alternative methods for tackling the problem [5–7] – these two directions have been summarized in [8]. Moreover, in Table 1 we present the overview of approaches for tackling the influence maximization problem.

Following the latter of the above-mentioned research directions, recently another line of research in this area is starting to be explored, as genetic algorithms shown their efficiency in multiple application areas already, such as: scheduling[9], engineering [10,11], path planning [12,13], data mining [14], management [15], loan portfolio optimization [16,17], biochar yield prediction [18,19] or wireless sensor networks lifetime extension [20,21]. The comparison of different evolutionary-based optimization algorithm can be found in [22,23]. Recently these are also being considered as prospective candidates in social network analysis as well. Some of the examples of their applications in online social networks analysis include community detection [24,25] or link prediction [26]. Recently, genetic algorithms are also starting to show their extensive capabilities in diffusion processes, yet the work so far can be considered in its early days and these initial attempts of their use in the area of influence maximization are described below.

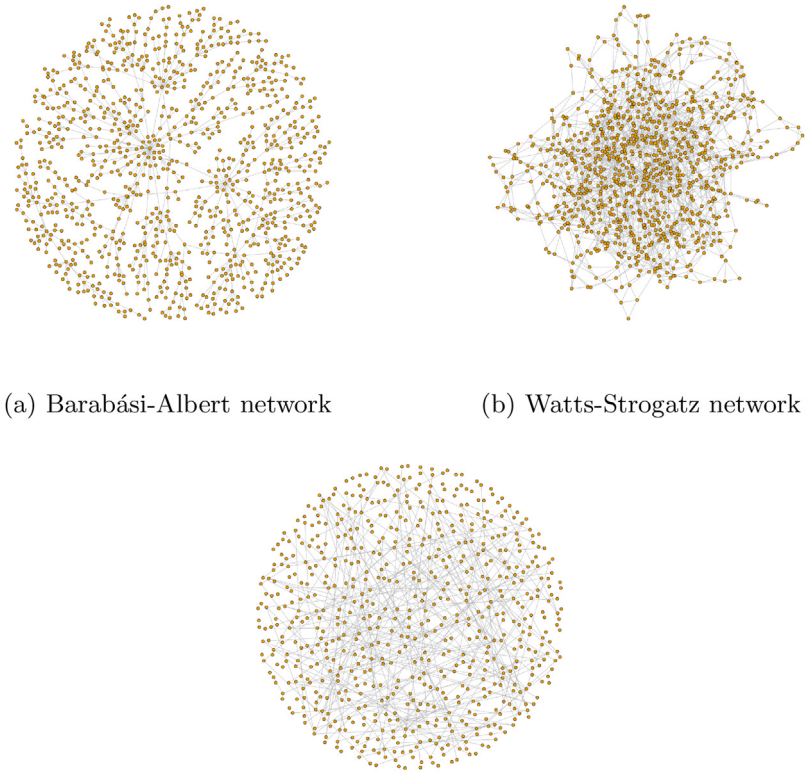
Table 1
The overview of methods for seed selection in social networks.

Method	Short description	Reference
Greedy	Iterative approach building the seed set by picking the nodes that together with already chosen ones provide maximum spread	[1,35]
Greedy optimization	Focus on optimizing the greedy algorithm by taking advantage of the submodulatity and monotonicity properties	[5,36,4]
Data-driven	Combining information about structural properties of nodes and past propagations	[36,37]
Structural	Taking into account the structural properties of nodes and links	[38,39,6]
Sequential seeding	Distributing the seeding over iterations in order to benefit from natural activations of nodes	[40–42]
Evolutionary	Early attempts of using genetic algorithm for influence maximization	[28,30]
Temporal seeding	Seed selection for time-varying networks	[43,44]

One of the first works that considered the use of genetic algorithms in the area of diffusion processes was not specifically designed for influence maximization, but for modeling the diffusion processes [27]. In this work, authors showed that it is possible to build a model of the diffusion process based on the evolutionary algorithm that can adequately represent this process found in the real world. The way of tackling the problem of influence maximization by using genetic algorithm was presented as a concept in [28]. Next, in [29] an approach for combining the greedy method with genetic algorithm in such a way that the modified greedy method improves the local search for the optimal solution for each crossover operation. Another approach, presented in [30] investigates how much gain can be achieved when the computations take place in GPGPU environment when comparing the evolutionary and greedy algorithms. The GPGPU approach is a one that belongs to the SIMD (single instruction multiple data) class in Flynn's computer architectures' taxonomy [31]. This means that in a single cycle one instruction is executed on multiple data. In the case of GPUs, the instructions are being executed on matrices, as previously this was the basic approach for filling the shapes with textures, light etc. in the case of computer graphics. Some years ago the GPUs started to be used for computations since many problems can be represented as operations on matrices. And, in the case of complex networks, matrices are also widely used as a way of representing the adjacency, allowing to increase the speed of computations compared to SISD or MIMD typical for single or multi-core CPUs as multiple matrices can be computed at once by thousands of GPU cores [32,33].

The results presented in [30] show a superiority of the evolutionary algorithm, both in terms of time and quality, yet some limitations were also identified, e.g., the maximum size of the network that can be processed. An interesting approach for considering how to generate the initial population and how to mutate samples was presented in [34] as G^2A approach. Here, the degree of the nodes is a non-varying indicator of spreading potential of vertices that can be taken into account in the genetic algorithm.

This work can be considered as a significant extension of the above-mentioned works in a couple of directions. Firstly, as the evolutionary algorithm method used in [30] imposed the limits on network size, here it is shown that these limits may be overcome by modifying the representation of the network. Secondly, and more importantly, in this work, an extensive investigation of the influ-



(a) Barabási-Albert network (b) Watts-Strogatz network

(c) Erdős-Rényi network

Fig. 1. Visualizations of the artificial networks used for experiments.

ence of genetic algorithm parameters on the results was performed, especially for transfer learning purposes. This leads to indications how volatile the results can be when applying the parameters for a genetic algorithm that were not computed based on a destination network. Since it has already been shown in [30], this work does not cover the comparison between evolutionary algorithm and other heuristics, but it focuses on an in-depth investigation of the performance of the evolutionary algorithm.

3. Experimental setup

The experiments were performed with both the real world and artificial networks using three networks models. Many parameters sets for the evolutionary algorithm were tested, and each test case was repeated and its results averaged. Finally, some credibility tests were conducted in order to check the results cohesion over different datasets. This section presents details of conducted experiments.

3.1. Datasets

For the experimental part, the real world datasets were chosen for obtaining representative results that can provide general conclusions being free of any dataset-specific overfitting. For that matter experiments were performed using both: the real life and randomly generated social networks following popular network models. The size of all of them was limited, in a way described later on, in order to reliably test as many parameters sets as possible within a reasonable time frame.

3.1.1. Real world networks

In order to provide reality consistent results, three of the real world social networks were used:

- Enron [45];
- Digg [46];
- Facebook wall posts [47].

Both *Enron* and *Digg* datasets contain the log files of messages being sent among employees of the Enron company and Digg social news website users, respectively. First of them contains 87,273 vertices (users) and 1,148,072 edges (messages), whilst the second – 30,398 vertices (users) and 87,627 edges (messages).

Facebook wall posts dataset is a subset of messages published on users' mutual profile pages, so-called *walls*. This set includes 46,952 vertices (users) and 876,993 edges (wall posts).

3.1.2. Artificial networks

Apart from the use of real world networks, three social networks models were used to create artificial undirected networks. The purpose of their use is to show that the results generalize on artificial networks following some standardized models, namely:

- Barabási-Albert (BA) model [48];
- Watts-Strogatz (WS) model [49];
- Erdős-Rényi (ER) model [50].

For each model, there were ten instances of artificial networks created. Each model introduces different parameters important in terms of the form of the network. The parameters were as follows:

- for the Barabási-Albert model the probability of attachment equals 1;
- for the Watts-Strogatz model the rewiring probability equals 0.001;
- for the Erdős-Rényi model the probability equals 0.05.

The visualizations of all artificial networks used for experiments are presented in Fig. 1.

3.2. Computation limitations

One of the disadvantages of the authors' previous approach [30] was the data representation. There, the$N$ by N matrix needed to be created in order to represent the influence values among$N$ nodes. That approach resulted in exponentially growing requirements regarding both RAM and the graphics card's memory when increasing the size of the network. This time authors decided to use the sparse matrix representation as shown in [51]. This approach led to increasing the maximum network size that can be handled by the setup used at the time from 1400 nodes to over 10,000. Further memory management improvements allowed networks of up to 23,000 nodes to be analyzed. Given that the implementation required a separate thread for each node of each individual, and the thread number limits of CUDA, the 23,000 was maximum possible for$N/8$ (maximum tested) number of individuals. When setting the number of individuals to one, the next limit reached was the amount of RAM memory used. At 40,000 nodes it peaked at 5.5GB which was close enough to the amount mounted (6GB) to call it a safe maximum for that setup. When testing full Enron network (over 87,000 nodes) and allowing the process to take over the swap memory it peaked at 29,4GB of total RAM memory and only 147 MB of the GPU memory, which is the equivalence of 1.3% total GPU memory of the card mounted. Nevertheless, the sizes of the networks tested needed to be limited to sizes far from those limits due to the fact that when increasing the network size, the computation time emerges rapidly. In order to perform as many parameters sets covering tests as possible in a reasonable time, it was inevitable to limit those networks' sizes. In the case of this experimental setup, the limit for the number of nodes was arbitrarily set to 1000 for all datasets.

3.3. Social influence model and influence maximization

The model chosen as a model for social influence was linear threshold model developed by Mark Granovetter in late 1970s [52]. This model assumes that surrounding individuals impose influence on a person that can be quantified and represented as influence weights. Moreover, the considered individual has its own threshold that, when exceeded by those summed influence of its influenced neighbors, also makes that individual influenced. The process is run iteratively in the network until no further activations are possible. In this setting, a seed set is a set of initially influenced individuals in the network that result in a given total number of influenced individuals at the end of this process. The influence weights in our settings were set accordingly to the following equation:

$$w_{ij} = \frac{\sum a_{ij}}{\sum a_j}, \tag{1}$$

where w_{ij} is the influence weight from node i to node j, $\sum a_{ij}$ is the number of all actions (messages, wall posts, etc.) from node i to node j and $\sum a_j$ is the total number of interactions of all node j's neighbors with node j. This can be interpreted in the way that the more interactions a given node has with another node, the higher its influence on that node is. The influence threshold of all nodes has been uniformly set to 0.5 meaning that in order to influence a node the weighted sum of its incoming edges has to exceed that value. The number of seeds in all experiments was set to be equal to 50 – an equivalence of 5% of the network. Additional experiments were performed in order to establish the correlation between results obtained for different sized seed sets. Please also note that despite the networks were imported as undirected, the influence weights following this definition already impose directions.

As it was shown in Section 2, many different approaches to finding best seed set exist, since the analytic solution is an NP-hard problem. In this work, we focus on the evolutionary algorithm as a mean for finding prospective candidate seed sets evaluating also the influence of that algorithm's parameters on the outcome of the process.

Table 2
Evolutionary algorithm's parameters values tested.

Group size	Nr of individuals	Crossover ratio	Mutation potency	Mutation ratio
10	$N/12$	0.6	0.001	0.7
20	**N/10**	**0.7**	**0.01**	0.8
30	$N/8$	0.8	0.1	**0.9**

3.4. Evolutionary algorithm

One of the main parts of this work was to examine whether one can say that some parameters sets of the evolutionary algorithm are generally better than others giving consistently higher results on the number of influenced nodes over many datasets. There are a couple of important parts that the evolutionary algorithm consists of: the initial population generation, selection, crossover, and mutation. Each part relies on some parameters and has different possible implementations. Initially, the population is generated based on the number of individuals parameter. Each individual created contains a randomized seed set (being 5% of the network in this study) – see Algorithm 2. Next, the evolutionary part begins, where selection, crossover, and mutation phases are executed in a loop (called generation) until some limit is reached – here, 6 s for real-world networks and 1.5 s for those that have been generated. When the limit is reached, the best individual's spreading capability (number of activated nodes) is saved – see Algorithm 1.

There are many possible selection phase approaches. Based on previous experience authors decided to use the tournament approach, in which there are groups of individuals randomized and their members are competing to advance to the next generation. Authors decided to advance only the best (most influential) individual out of the group to the next generation, for details see Algorithm 3. In the crossover procedure, individuals are exchanging part of their "genetic code" within the population. Here, those have been exchanging 50% of their nodes. The crossover procedure depends on the crossover ratio parameter, which determines the chance of the crossover between individuals taking place. This procedure has been described in Algorithm 4. During the mutation phase, some of the individuals are mutating – changing their "genetic code". The mutation potency is a parameter describing a fraction of the individual being swapped with randomly chosen nodes, and the mutation ratio is a chance of it happening – see Algorithm 5.

Authors decided to use three values for each of the aforementioned parameters and conducted tests over all possible combinations (243) of those values (presented in Table 2). The values itself were based on the results obtained previously [30] (marked as bold). It was a natural continuation into the best parameters values determination for the given problem.

As far as the algorithm's computation time is concerned, it was arbitrarily set to 6 s for real-world datasets and 1.5 s for generated networks. The difference in the computation time provided even

more diverse environments in which the parameters sets could be tested and their transferability showed. In those time spans, the number of generations occurred ranged between 30 and 300 (depending on the dataset being used and the parameters tested), see Section 4.1 for details.

Algorithm 1. Evolutionary algorithm

```
procedure ALGORITHM'S BODY
  for all possible combinations of parameters values (243) do
    for all tests (10) do
      procedure POPULATION INITIALIZATION
      while Time limit not reached do
        procedure SELECTION
        procedure CROSSOVER
        procedure MUTATION
    Average the results of 10 tests
    Save the final result of the given parameters set
```

Algorithm 2. Population initialization

```
procedure INITIAL POPULATION RANDOMIZATION
  Create empty population P
  while size of P < numberOfIndividuals do
    Create empty individual I
    while size of I < seedSetSize (50) do
      Assign randomized node id to individual I
    end while
    Add individual I to population P
```

Algorithm 3. Selection

```
procedure SELECTION
  for all individuals in population do
    Calculate the influence spreading capabilities (effect of
  activating nodes defined by the individual) on GPU]]
  Create empty population P2
  while Size of P2 =/= size of P do
    Randomize groupSize individuals
    Choose the best one (biggest influence) individual and
  add it to P2
  Replace P's individuals with P2's individuals
```

Algorithm 4. Crossover

```
procedure CROSSOVER
  for all individuals in population do
    Randomize a number between 0 and 1
    if Randomized number < crossoverRatio then
      if Any individual in the waiting list then
        Divide current individual in half
        Divide awaiting individual in half
        Swap first halves (50% of nodes) of those individuals
        Clear the waiting list
      else
        Add individual to the waiting list
```

Algorithm 5. Mutation

```
procedure MUTATION
  for all individuals in population do
    Randomize a number between 0 and 1
    if Randomized number < mutationRatio then
      while Nr. of mutations < mutationPotency * seedSetSize(50) do
        Randomize a node within the individual to be replaced
        Randomize replacement node
        Exchange the node with the replacement
```

3.5. Results averaging

For each network, there have been 243 test cases, which equals the number of possible parameters' combinations. For each test case, there have been ten independent runs of the genetic algorithm performed in which the best possible individual (seed set) was looked for. The influential capabilities of an individual were interpreted as the total number of activated nodes in the influence spread process. After running the process of the influence spread the best performing individuals were saved. As a result, each dataset provided data containing 2430 values used to evaluate each of the parameters sets by averaging over ten entries each. The sets of 243 results have later on been used for both: (i) establishing the best performing parameters sets among all networks, (ii) to calculate the correlation coefficients between them, further described in detail. It needs to be underlined that in the case of generated networks, there were actually ten instances created for each model and the results presented later on are averaged over those ten instances' final results.

3.6. Results credibility and cohesion tests

It was decided to use some measurements in order to ensure the cohesion of results over all of the datasets. For that matter, the Pearson's correlation coefficient was calculated [53] comparing the vectors containing all 243 final results for each pair of the datasets tested. Then, those averaged results were also used to create unique parameters sets ranks for each dataset. Then, the Spearman's correlation coefficient [54] was calculated to obtain the dependency of those rankings.

3.7. Hardware and software

As the computation performed relies on the GPU computation, the CUDA-capable GPU was used. The main hardware and software components of the workstation used for experiments are listed below:

- NVIDIA 1080 Ti (11GB GDDR5, 3584 CUDA cores)
- 2 x Intel Xeon Processor E5640 (12M Cache, 2.66 GHz, 5.86 GT/s)
- 6 GB of RAM
- Ubuntu Linux with kernel 4.4.0-137-generic
- NVIDIA CUDA runtime and drivers version 10.0

In order to provide other researchers with the ability to reproduce the results or conduct similar experiments based on the introduced approach, we published the source code used for the experimental part on the GitHub platform.[1]

4. Results and discussion

This section presents all performed experiments as well as an interpretation of the values obtained. Moreover, the way that the results were changing over time is being taken into consideration. Also, the measuring methods for the data coherency were used and described below. Finally, the standard deviation over multiple tests was investigated.

4.1. Result over time

One of the aspects of the way that the evolutionary algorithm performs that has been intended to be tested by the authors is the change in the result obtained over time. This is also one of the main advances when compared to other approaches, like the greedy algorithm, that in this case the result may be presented in any moment during the simulation, just probably not very high for the short time given. The result is then going to be better and better over time and the researcher is not constrained with any minimal time limit needed to find a solution. The way that the result changes over time in the approach used by the authors is presented in Figs. 2 and 3 . Clearly, high results are being obtained early in the process, which is one of the characteristics typical for genetic algorithms. Then, the maximal influence value is increasing over the next generations

[1] https://github.com/Mhal007/evolutionaryinfluencespreadsparse.

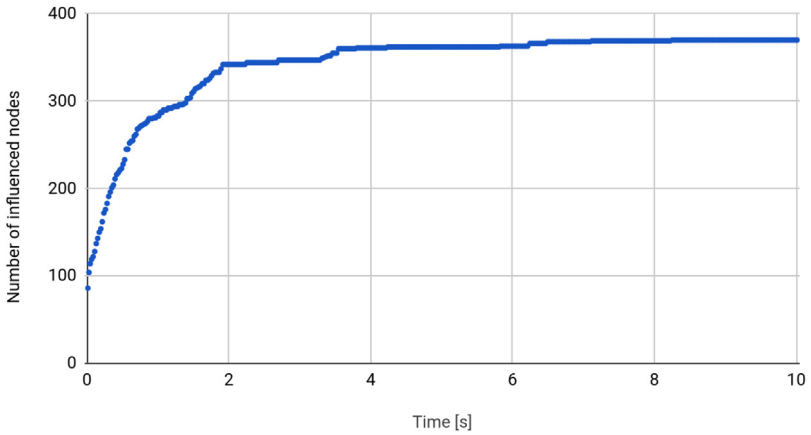


Fig. 2. The dependency between the execution time of the evolutionary algorithm and the capabilities of the best individual found – over one of the real-world networks – Facebook. Computation time in case of those networks was limited to 6 s.

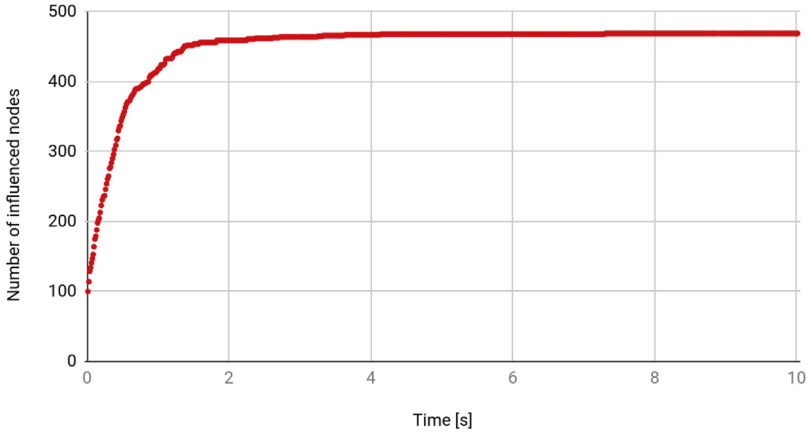


Fig. 3. The dependency between the execution time of the evolutionary algorithm and the capabilities of the best individual found – over one of the generated networks – BA. Computation time in case of those networks was limited to 1.5 s.

when it reaches its peak in less than 10 s. No further optimization was achieved in this trial. That kind of analysis was conducted before starting the experimental phase of the research in order limit the test cases in a moment that seems reasonable, namely, when a high result is already reached but it is still far from being an optimal solution. Authors' intention was to discover those parameters sets that provide satisfactory results in a relatively, to the nature and complexity of the problem, short time. Based on this preliminary computation, the limit has been set to 6 s for real-world datasets and to 1.5 s for generated networks.

4.2. Best parameters sets

The methodology of finding best parameters combinations was to calculate the average position of results obtained when using a given set of parameters over all datasets, then creating a ranking where the higher average position the better. Those findings were presented in Fig. 3 . The last column presents the aforementioned average position obtained. Clearly, the best group size, among values tested, seems to be equal to 30. No other value of this parameter is present in the top results. Two most efficient number of individuals were 1/10th and 1/12th of the network size, here 100 and 83 individuals, respectively (after rounding). Generally speaking, it would seem like the more individuals the better the result, however, increasing number of individuals influences the computation time per generation, resulting in fewer generations performed overall in a given time frame. In this case, the maximum value was not really the most efficient one. As for the crossover ratio, it is often recommended to use a value of this parameter close to 0.7. These results show that all of the three values tested are able to provide satisfactory results, although the best parameters set contains 0.6 as its value. Contradictory, the mutation ratio values stay uniform across best results at 0.9, only one set containing 0.8. Quite remarkable, that such high mutation ratio is giving the best results when it is often recommended to keep this value relatively small. The reason behind it is the use of the additional parameter - mutation potency - with very small values. In the case of this experiment, the mutation potency values of 0.01 and 0.001 influenced the mutation phase in the same way, namely, mutating exactly one node of the individual. Overall, the combination of high mutation ratio with low mutation potency provides multiple minor

Table 3
Best parameters combinations among all datasets.

Group size	Nr. of indiv.	Cross. ratio	Mut. ratio	Mut. potency	Avg position
30	83	0.6	0.9	0.01	10
30	100	0.8	0.9	0.01	10
30	100	0.7	0.9	0.001	13
30	100	0.6	0.9	0.001	14
30	83	0.8	0.9	0.001	14
30	83	0.8	0.9	0.01	14
30	100	0.6	0.9	0.01	14
30	83	0.7	0.9	0.001	15
30	100	0.7	0.9	0.01	20
30	100	0.7	0.8	0.001	21

Table 4
Pearson's correlation coefficient for the values obtained among different datasets.

	Digg	Enron	Facebook	BA	ER	WS
Digg	–	0.335	0.970	0.983	0.981	0.973
Enron	0.335	–	0.440	0.362	0.369	0.239
Facebook	0.970	0.440	–	0.981	0.983	0.941
BA	0.983	0.362	0.981	–	0.998	0.982
ER	0.981	0.369	0.983	0.998	–	0.977
WS	0.973	0.239	0.941	0.982	0.977	–

Table 5
Spearman's correlation coefficient for the ranks obtained among different datasets.

	Digg	Enron	Facebook	BA	ER	WS
Digg	–	0.127	0.830	0.929	0.933	0.941
Enron	0.127	–	0.212	0.116	0.102	0.078
Facebook	0.830	0.212	–	0.859	0.858	0.842
BA	0.929	0.116	0.859	–	0.981	0.975
ER	0.933	0.102	0.858	0.981	–	0.965
WS	0.941	0.078	0.842	0.975	0.965	–

Table 6
Pearson's correlation coefficient among different number of seeds.

Faceboook	3% seeds	5% seeds	7% seeds
3% seeds	–	0.977	0.977
5% seeds	0.977	–	0.993
7% seeds	0.977	0.993	–

mutations which seems to be the best combination for obtaining the best overall results.

4.3. Results coherency

In order to ensure that the obtained results are reliable across all datasets, authors calculated the Pearson's and Spearman's correlation coefficients presented in Tables 4 and 5 , respectively. Usually, the value above 0.4 is interpreted as proof of a statistically significant correlation between results. This value was easily obtained in most cases. The only dataset struggling with transcending this threshold was the Enron dataset. The highly probable reason for low results between Enron and other datasets is the fact that for this particular network the results obtained were really high – ranging between 917 and 947, where 950 is the maximum possible result. Spreading the influence within this network seems to have been so trivial task to solve, that the advantages of certain parameters sets of the evolutionary algorithm did not have a chance to show their potential. Apart from this dataset, the correlation values are very high, ranging from 94.1% all the way up to 99.8% for the Pearson's coefficient and 83% to 98.1% in case of the Spearman's coefficient.

The highest correlation occurred between the BA and ER networks for both approaches. All in all, the results obtained in the Pearson's test were consistent with those obtained with the use of the Spearman's coefficient. Some differences were present due to the slightly different approach between the two, but they were mostly consistent.

An additional study was performed in order to find if different sized seed sets can meaningfully influence the results obtained. For that matter there were tests covering all 243 parameters sets carried out, with ten runs for each set averaged, to obtain final results. This experiment was repeated for three seed set sizes – 3%, 5% and 7% of the network size. In all cases, the Facebook network have been used. As can be seen in Tables 6 and 7 , the results were uniformly highly correlated with each other (over 97% for Pearson's and 77% for Spearman's coefficients), meaning the results obtained in this study are also transferable to studies based on other seed set sizes.

Table 7
Spearman's correlation coefficient among different number of seeds.

Facebook	3% seeds	5% seeds	7% seeds
3% seeds	–	0.776	0.774
5% seeds	0.776	–	0.859
7% seeds	0.774	0.859	–

Table 8
The mean value and standard deviation of the number of influenced nodes for a chosen combination of evaluated parameters (Digg network).

Parameter	Value
Group size	30
Number of individuals	83
Crossover ratio	0.6
Mutation potency	0.01
Mutation ratio	0.9
Mean number of influenced nodes	*385.6*
Standard deviation	*8.027*

4.4. Results standard deviation

As mentioned before, each parameters' set was tested with ten trials over a given dataset. Those results were then averaged for the purpose of further calculations. Apart from the final, averaged value, it seems worth mentioning how those results were spread amongst all trials. And that highly depends on the parameters set being tested and the dataset that the test is being performed on. An example of the values obtained and their standard deviation is presented in Table 8. In most cases the standard deviation was not particularly significant, however, the repeated tests were needed in order to obtain more reliable results due to the fact that the evolutionary algorithm is non-deterministic.

5. Conclusions and future work

In this work, we investigated the influence of the evolutionary algorithm parameters on the result measured as the number of influenced nodes. Combined with our earlier work on the topic in which we compared greedy algorithm against this evolutionary approach, the results clearly prove that the evolutionary method holds an important position being fast and accurate, especially when compared to the greedy approach. In terms of searching the parameters' space, the best found parameters sets were presented that can be used in the future. Furthermore, the Pearson's and Spearman's correlation coefficients' values proved that these sets performed well across all of the datasets that have been tested. The main conclusion is not that those particular sets are generally the best, surely a followup paper examining a wider set of parameters, with the use of more powerful equipment and longer lasting experiments could lead to finding some other, better performing sets. The main point is that it is surely doable to obtain the parameters' values that are constantly giving better results, even for datasets with various sizes, complexity, a model they are based on, or computation time restraint. Also, the sparse data representation implementation opened up a possibility of performing simulations on significantly bigger datasets. Furthermore, the analysis of the results' change over time has shown that the evolutionary approach may be the best choice in case of dynamic networks, or when the result is needed quickly, as it provides quality results in no time, when compared to the standard non-heuristic analysis, where working with big datasets requires a huge amount of computation time to obtain any result at all. Here, the evolutionary approach achieves high results very fast and the more time is spent the better result is obtained while allowing to provide results at any time and to stop the computation at any particular moment.

In the future, the main focus should be pointed towards investigating whether some heuristics can be used in order to test a wider parameters space allowing to determine better parameters values sets maximizing the influence spread even further. Apart from that, an interesting research direction arises in which a streaming graph scenario would be considered. This will allow the proposed method to be even more suitable for time evolving online social networks.

Acknowledgments

This work was supported by the National Science Centre, Poland, projects no. 2015/17/D/ST6/04046 (RM) and 2016/21/B/ST6/01463 (MW) as well as by the European Unions Horizon 2020 research and innovation programme under grant agreement no. 691152 (RENOIR) and the Polish Ministry of Science and Higher Education fund for supporting internationally co-financed projects in 2016–2019 (agreement no. 3628/H2020/2016/2).

References

[1] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 137–146.
[2] T. Maehara, H. Suzuki, M. Ishihata, Exact computation of influence spread by binary decision diagrams, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 947–956.
[3] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 420–429.
[4] A. Goyal, W. Lu, L.V. Lakshmanan, Celf++: optimizing the greedy algorithm for influence maximization in social networks, in: Proceedings of the 20th International Conference Companion on World Wide Web, ACM, 2011, pp. 47–48.
[5] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 199–208.
[6] W. Chen, C. Wang, Y. Wang, Scalable influence maximization for prevalent viral marketing in large-scale social networks, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 1029–1038.
[7] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, K. Xie, Simulated annealing based influence maximization in social networks, AAAI (2011).
[8] R. Michalski, P. Kazienko, Maximizing social influence in real-world networks: the state of the art and current challenges, in: Propagation Phenomena in Real World Networks, Springer, 2015, pp. 329–359.
[9] A.A.R. Hosseinabadi, J. Vahidi, B. Saemi, A.K. Sangaiah, M. Elhoseny, Extended genetic algorithm for solving open-shop scheduling problem, Soft Computing (2018) 1–18.
[10] D. Dasgupta, Z. Michalewicz, Evolutionary Algorithms in Engineering Applications, Springer Science & Business Media, 2013.
[11] T. Jayabarathi, T. Raghunathan, A. Gandomi, The bat algorithm, variants and some practical engineering applications: a review, in: Nature-Inspired Algorithms and Applied Optimization, Springer, 2018, pp. 313–330.
[12] A. Tharwat, M. Elhoseny, A.E. Hassanien, T. Gabel, A. Kumar, Intelligent Beziér curve-based path planning model using chaotic particle swarm optimization algorithm, Cluster Computing (2018) 1–22.
[13] M. Elhoseny, A. Tharwat, A.E. Hassanien, Bezier curve based path planning in a dynamic field using modified genetic algorithm, Journal of Computational Science 25 (2018) 339–350.
[14] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C.A.C. Coello, A survey of multiobjective evolutionary algorithms for data mining: Part I, IEEE Trans. Evolutionary Computation 18 (1) (2014) 4–19.
[15] V. Nissen, Applications of evolutionary algorithms to management problems, in: Innovative Research Methodologies in Management, Springer, 2018, pp. 211–235.
[16] N. Metawa, M. Elhoseny, M.K. Hassan, A.E. Hassanien, Loan portfolio optimization using genetic algorithm: a case of credit constraints, in: 12th International Computer Engineering Conference (ICENCO), IEEE, 2016, pp. 59–64.
[17] N. Metawa, M.K. Hassan, M. Elhoseny, Genetic algorithm based model for optimizing bank lending decisions, Expert Systems with Applications 80 (2017) 75–82.
[18] M.A. El Aziz, A.M. Hemdan, A.A. Ewees, M. Elhoseny, A. Shehab, A.E. Hassanien, S. Xiong, Prediction of biochar yield using adaptive neuro-fuzzy inference system with particle swarm optimization, in: PowerAfrica, 2017 IEEE PES, IEEE, 2017, pp. 115–120.
[19] A.A. Ewees, M.A. El Aziz, M. Elhoseny, Social-spider optimization algorithm for improving ANFIS to predict biochar yield, in: 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, 2017, pp. 1–6.
[20] M. Elhoseny, A. Tharwat, A. Farouk, A. Hassanien, K-coverage model based on genetic algorithm to extend WSN lifetime, IEEE Sens Lett 1 (4) (2017) 1–4.
[21] X. Yuan, M. Elhoseny, H.K. El-Minir, A.M. Riad, A genetic algorithm-based, dynamic clustering method towards improved WSN longevity, Journal of Network and Systems Management 25 (1) (2017) 21–46.
[22] E. Elbeltagi, T. Hegazy, D. Grierson, Comparison among five evolutionary-based optimization algorithms, Advanced Engineering Informatics 19 (1) (2005) 43–53.
[23] P. Civicioglu, E. Besdok, A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artificial Intelligence Review 39 (4) (2013) 315–346.
[24] C. Pizzuti, Ga-net: a genetic algorithm for community detection in social networks, in: PPSN, Springer, 2008, pp. 1081–1090.
[25] W.A. Hariz, M.F. Abdulhalim, et al., Improving the performance of evolutionary multi-objective co-clustering models for community detection in complex social networks, Swarm and Evolutionary Computation 26 (2016) 137–156.
[26] C.A. Bliss, M.R. Frank, C.M. Danforth, P.S. Dodds, An evolutionary algorithm approach to link prediction in dynamic social networks, Journal of Computational Science 5 (5) (2014) 750–764.
[27] M. Lahiri, M. Cebrian, The genetic algorithm as a general diffusion model for social networks, AAAI (2010).
[28] J. Guo, Y. Liu, J. Shen, Z. Wei, J. Lv, Influence maximization algorithm based on genetic algorithm, Journal of Computational Information Systems 10 (21) (2014) 9255–9262.
[29] C.-W. Tsai, Y.-C. Yang, M.-C. Chiang, A genetic newgreedy algorithm for influence maximization in social network, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2015, pp. 2549–2554.
[30] M. Weskida, R. Michalski, Evolutionary algorithm for seed selection in social influence process, in: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, 2016, pp. 1189–1196.
[31] M.J. Flynn, Some computer organizations and their effectiveness, IEEE Transactions on Computers 100 (9) (1972) 948–960.
[32] X. Yang, S. Parthasarathy, P. Sadayappan, Fast sparse matrix–vector multiplication on GPUS: implications for graph mining, Proceedings of the VLDB Endowment 4 (4) (2011) 231–242.
[33] P. Harish, P. Narayanan, Accelerating large graph algorithms on the GPU using CUDA, in: International Conference on High-performance Computing, Springer, 2007, pp. 197–208.
[34] P. Krömer, J. Nowaková, Guided genetic algorithm for the influence maximization problem, in: International Computing and Combinatorics Conference, Springer, 2017, pp. 630–641.
[35] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, Theory of Computing 11 (4) (2015) 105–147.
[36] A. Goyal, F. Bonchi, L.V. Lakshmanan, A data-based approach to social influence maximization, Proceedings of the VLDB Endowment 5 (1) (2011) 73–84.
[37] N. Barbieri, F. Bonchi, G. Manco, Topic-aware social influence propagation models, in: 2012 IEEE 12th International Conference on Data Mining (ICDM), IEEE, 2012, pp. 81–90.
[38] Y. Chen, S. Chang, C. Chou, W. Peng, S. Lee, Exploring community structures for influence maximization in social networks, The 6th SNA-KDD Workshop on Social Network Mining and Analysis Held in Conjunction with KDD, vol. 12 (2012) 1–6.
[39] Y.-C. Chen, W.-Y. Zhu, W.-C. Peng, W.-C. Lee, S.-Y. Lee, Cim: community-based influence maximization in social networks, ACM Transactions on Intelligent Systems and Technology (TIST) 5 (2) (2014) 25.
[40] J. Jankowski, P. Bródka, P. Kazienko, B.K. Szymanski, R. Michalski, T. Kajdanowicz, Balancing speed and coverage by sequential seeding in complex networks, Scientific Reports 7 (1) (2017) 891.
[41] J. Jankowski, B.K. Szymanski, P. Kazienko, R. Michalski, P. Bródka, Probing limits of information spread with sequential seeding, Scientific Reports 8 (1) (2018) 13996.
[42] J. Jankowski, M. Waniek, A. Alshamsi, P. Bródka, R. Michalski, Strategic distribution of seeds to support diffusion in complex networks, PLoS ONE 13 (10) (2018) e0205130.
[43] J. Jankowski, R. Michalski, P. Kazienko, Compensatory seeding in networks with varying availability of nodes, in: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ACM, 2013, pp. 1242–1249.
[44] R. Michalski, T. Kajdanowicz, P. Bródka, P. Kazienko, Seed selection for spread of influence in social networks: temporal vs. static approach, New Generation Computing 32 (3-4) (2014) 213–235.
[45] B. Klimt, Y. Yang, The enron corpus: a new dataset for email classification research, in: Machine learning: ECML 2004, Springer, 2004, pp. 217–226.
[46] M. De Choudhury, H. Sundaram, A. John, D.D. Seligmann, Social synchrony: predicting mimicry of user actions in online social media, in: International Conference on Computational Science and Engineering, 2009, CSE'09, vol. 4, IEEE, 2009, pp. 151–158.

[47] B. Viswanath, A. Mislove, M. Cha, K.P. Gummadi, On the evolution of user interaction in Facebook, in: Proceedings of the 2nd ACM workshop on Online Social Networks, ACM, 2009, pp. 37–42.
[48] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[49] D.J. Watts, S.H. Strogatz, Collective dynamics of small-world networks, Nature 393 (6684) (1998) 440–442.
[50] P. Erdos, A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci. 5 (1960) 17–61.
[51] X. Liu, M. Li, S. Li, S. Peng, X. Liao, X. Lu, Imgpu: Gpu-accelerated influence maximization in large-scale social networks, IEEE Transactions on Parallel and Distributed Systems 25 (1) (2014) 136–145.
[52] M. Granovetter, Threshold models of collective behavior, American Journal of Sociology 83 (6) (1978) 1420.
[53] K. Pearson, Note on regression and inheritance in the case of two parents, Proceedings of the Royal Society of London 58 (1895) 240–242.
[54] C. Spearman, The proof and measurement of association between two things, The American Journal of Psychology 15 (1) (1904) 72–101.

Michał Weskida is a graduate of Computer Science at the Faculty of Computer Science and Management at Wrocław University of Science and Technology, Wrocław, Poland. In his master thesis he investigated the possibility of using evolutionary algorithm for influence maximization. His work on that has been announced Best Paper at the 6th Workshop on Social Network Analysis in Applications (SNAA 2016). He is currently combining his academic work of improving social networks analysis with the use of artificial intelligence applications, with working in the IT business, providing web-oriented solutions to customers around the Globe.

Radosław Michalski is an Assistant Professor at the Department of Computational Intelligence at Wrocław University of Science and Technology, Wrocław, Poland. His research areas cover, but are not limited to: social influence, diffusion processes in complex networks, as well as machine learning. He co-authored over 50 publications in these areas, ten of which were published in influential journals.