Influence Maximization in Social Networks with Genetic Algorithms

Doina Bucur[1] and Giovanni Iacca[2]

[1] Johann Bernoulli Institute, University of Groningen
Nijenborgh 9, 9747 AG Groningen, The Netherlands
d.bucur@rug.nl
[2] INCAS[3]
Dr. Nassaulaan 9, 9401 HJ, Assen, The Netherlands
giovanniiacca@incas3.eu

Abstract. We live in a world of social networks. Our everyday choices are often influenced by social interactions. Word of mouth, meme diffusion on the Internet, and viral marketing are all examples of how social networks can affect our behaviour. In many practical applications, it is of great interest to determine which nodes have the highest influence over the network, i.e., which set of nodes will, indirectly, reach the largest audience when propagating information. These nodes might be, for instance, the target for early adopters of a product, the most influential endorsers in political elections, or the most important investors in financial operations, just to name a few examples. Here, we tackle the NP-hard problem of influence maximization on social networks by means of a Genetic Algorithm. We show that, by using simple genetic operators, it is possible to find in feasible runtime solutions of high-influence that are comparable, and occasionally better, than the solutions found by a number of known heuristics (one of which was previously proven to have the best possible approximation guarantee, in polynomial time, of the optimal solution). The advantages of Genetic Algorithms show, however, in them not requiring any assumptions about the graph underlying the network, and in them obtaining more diverse sets of feasible solutions than current heuristics.

Keywords: Social Network, Influence Maximization, Genetic Algorithm, Graph Theory, Combinatorial Optimization.

1 Introduction

Social networks are graphs of relationships natural to organized societies. Among humans, these relationships are the vehicle by which news, ideas, trends, advertising, or influence will spread, starting from an initial set of information owners. A process of influence spread is shown abstractly in Figure 1. In a social network where a graph edge $a \to b$ signifies a likelihood that user a will support b in any election (e.g., for the purpose of work-related promotions), the long-term promotion outcome will depend on the initial set of network participants who cast

a vote. Also, in a product-marketing network where the nodes are products and the edges $a \to b$ model the likelihood of clients buying product b after product a was bought, an advertiser can increase sales by specifically promoting that set of products which trigger the largest co-buying effect upon the rest of the network.

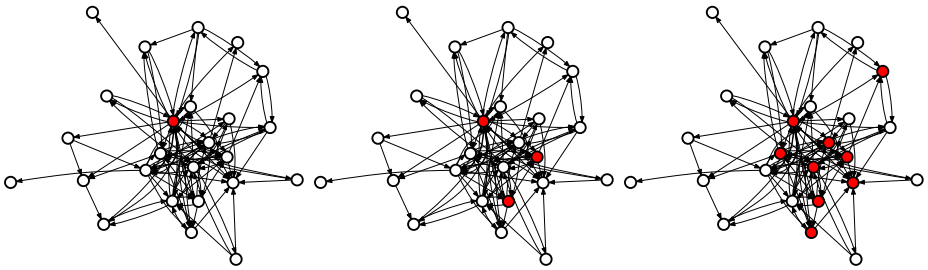


Fig. 1. Schematic spread of influence in a social network: three discrete steps in the state of the network starting from a single "seed" node. Nodes reached are drawn in red (grey in print). The information propagates via edges according to specific probabilistic spread models, each spread model particular to a type of network.

The precise dynamics by which the graph structure enables new information to spread depends on the nature of the social network: a directed graph edge $a \to b$ may simply model a fixed probability that information will be adopted by b if a has just done so; alternatively, the likelihood of influence propagation across the edge depends on other features of node b, such as its number of direct relationships with other nodes. Social sciences have studied a number of such probabilistic *propagation models* [1].

The open problem of *influence maximization* in a social network is the following: given the network graph G, a discrete-time formal propagation model M, and a numerical "budget" $k \geq 1$ of network nodes to be initial "seeds" of influence, calculate that set of k seed nodes which will have the largest global influence upon the network. The problem was initially formulated in [2], and was proven to be NP-hard for most propagation models [1].

In this work, we tackle the influence maximization problem by means of a Genetic Algorithm (GA) which uses simple genetic operators typically used in discrete optimization. We evaluate the Genetic Algorithm on two large, real-world network datasets from the SNAP repository [3] modelling: (1) the who-voted-whom network of 7115 Wikipedia users through a number of years, and (2) a snapshot of the Amazon product co-purchasing network, of 262111 products. We compare the results against three existing heuristics (all of a greedy nature, either based on graph theory, or on exhaustive incremental search), and also with randomly sampled solutions. We show that, even on very large networks, the GA is at least as good as some of the known heuristics, without using any

domain knowledge of the underlying graphs. Furthermore, the GA is able to find multiple diverse solutions with equally high network influence, suggesting that the fitness landscape of this problem may have a high level of multimodality.

The remainder of the paper is structured as follows. First, in the next section we briefly review the background concepts on influence propagation and the ways influence can be maximized, by using heuristics. Then, in Section 3 we describe the Genetic Algorithm used in our tests. Section 4 reports the experimental results and the related analysis. Finally, Section 5 concludes this work.

2 Background: network and propagation models

A social network is modelled as a directed graph G. For the purpose of studying the propagation of influence, at a time t each node in G is either *active* (i.e., has adopted the new information) or *inactive*. As will be seen in Section 2.1 below, in the propagation models we use, the set of active nodes in G increases monotonously until the propagation process ends.

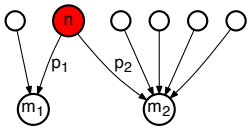
2.1 Modelling the propagation of influence: the cascade models

We consider a family of two classic, discrete-time influence-propagation models in social networks, known as "cascade" models [1], see Algorithm 1. In these discrete-time propagation models, the dynamics of information adoption is represented in individual steps. A given set A_0 of "seed" nodes start in the active state; these are the network entities which are originally targeted in the influence process. In the next time step $t \geq 1$, each node activated at time $t-1$ may activate some other nodes according to a probability given by the propagation model; in the case of cascade models, this probability of activation is attached to every edge, and is an independent random variable. The propagation ends at the time t when no new nodes were activated, and the set containing all nodes made active before t is the end result of the propagation process.

The simplest cascade model is the Independent Cascade, first studied in the marketing domain, as an attempt to understand the effects that personal word-of-mouth communication has upon macro-level marketing [4]. In the Independent Cascade, each newly activated node n will succeed in activating each currently inactive neighbour m with a fixed probability p, which is a global property of the system, and is thus equal across all edges $n \to m$; when node n has more than one neighbour, the attempts at activation are sequenced in arbitrary order. As an example, for the simple network in Figure 2, in which $A_0 = \{n\}$, the model has $p_1 = p_2 = p$ and, at time $t = 1$, nodes m_1 and m_2 are equally likely to become active. Given $p = 0.5$, the expected size of the set of active nodes at the end of the propagation process is 2; this count includes the seed node n itself.

The second cascade model, named Weighted Cascade, differs from the Independent Cascade in that it assigns non-uniform probabilities of activation to edges: an edge $n \to m$ has probability $\frac{1}{\text{in-degree}(m)}$ of activating m when n is itself

Fig. 2. Simple propagation example for the cascade models. With Independent Cascade, $p_1 = p_2$; with Weighted Cascade, $p_1 = 1/2$ and $p_2 = 1/5$.



active[3]. It holds that, unlike for the Independent Cascade model, the expected number of neighbours which will succeed in activating any node equals 1.

Algorithm 1 The **Cascade** family of propagation models. G is the network graph, A_0 the set of "seed" nodes, and $p(n \to m)$ the probability that information will reach across a directed graph edge $n \to m$. In the **Independent Cascade** model, $p(n \to m)$ is constant and equal to the input parameter p for all edges $n \to m$ in G. In the **Weighted Cascade** model, $p(n \to m)$ is instead equal to $\frac{1}{\text{in-degree}(m)}$.

```
1:  procedure CASCADE(G, A_0, p)
2:      τ ← False                                   ▷ τ: has the propagation ended?
3:      A ← A_0                  ▷ A: the set of active nodes after the propagation ended
4:      B ← A                        ▷ B: the set of nodes activated in the last time slot
5:      while not τ do
6:          nextB ← ∅
7:          for each n ∈ B do                ▷ only nodes in B will activate new nodes
8:              for each direct neighbour m of n in G, where m ∉ A, do
9:                  with probability p(n → m), add m to nextB
10:         B ← nextB
11:         A ← A ∪ B
12:         if B is empty then
13:             τ ← True
14:     return the size of A
```

2.2 Problem statement

The influence-maximization problem optimizes the choice of the seed nodes in set A_0. The *influence* of a given seed set A_0, denoted $\sigma(A_0)$, is the expected size of the set A of active nodes, $\mathbb{E}[|A|]$, obtained by a propagation model from Algorithm 1 after completion. Given a number $k \geq 1$, the problem asks to

[3] For any node n, we denote by *in-degree*(n) the number of edges incoming to n, and by *out-degree*(n) the number of edges outgoing from n. Unlike some of the related literature, which works with undirected rather than directed graphs, in our algorithms we make the distinction between the two degree counts explicit.

compute the optimum set A_0, where $|A_0| = k$, such that $\sigma(A_0) = \mathbb{E}[|A|]$ is maximized over all possible sets A_0 in the graph G.

For both models, the problem of calculating the optimal "seed" set is NP-hard. Further, we also know an approximation hardness result: estimating the optimal solution by a factor better than $1 - \frac{1}{e}$ (where e is the base of the natural logarithm) is also NP-hard [1]; this was proven by showing that it is at least as hard as the classical NP-hard problem of maximum coverage (i.e., determining those k sets of elements whose union has the maximum size). This approximation factor amounts to an approximation guarantee just above 63%, i.e., it is not possible to obtain in polynomial time a set A_0 whose influence is higher than 63% of the true optimum.

2.3 Existing heuristics. Their complexity and approximation guarantees

Approximation algorithms are used to compute best-effort solutions. In this paper, we use three of the existing approximation heuristics as a basis for comparison of performance: *General greedy* (due to its high complexity, only when feasible computationally), *High degree*, and *Single discount*. The Kempe et al. [1] greedy hill-climbing algorithm (referred to in this paper as the *General greedy* heuristic) is proven to approximate the solution to within a factor arbitrarily close to the approximation guarantee; we thus use *General greedy* as an optimality benchmark. This heuristic has the following logic: it starts with $A_0 = \emptyset$ and the given problem size k, and adds one node at a time to A_0 until $|A_0| = k$. A new node n from G, with $n \notin A_0$, is chosen to be added to A_0 if n maximizes $\sigma(A_0 \cup \{n\})$.

This means that all nodes in G, but not yet in A_0 must have their added influence evaluated by computing their $\sigma(A_0 \cup \{\cdot\})$. Note that, since the propagation models are stochastic, the evaluation of $\sigma(A_0 \cup \{\cdot\})$ in polynomial time will not be exact; the problem of exactly computing $\sigma(A_0)$ for any A_0 under the Independent Cascade model was proven #P-complete in Wang et al. [5]. However, one can approximate $\sigma(A_0)$ by simulating the propagation process a number of times, as the computational budget allows.

The heuristics which followed the *General greedy* method fell into two categories: (a) heuristics which preserve the approximation guarantee of *General greedy* while lowering its average-case (but not worst-case) complexity, and (b) heuristics of better complexity, but either no optimality guarantees or much weaker ones.

From the latter category of heuristics without any guarantees attached, Kempe et al. [1] also tested experimentally the performance of a *High-degree* (or degree centrality) heuristic, which adds nodes n to A_0 in order of decreasing out-degrees. They motivate this heuristic as being a standard greedy heuristic for networks. Chen et al. [6] designs degree-discount heuristics based on *High-degree*: if a node n is already in A_0 and there exists an edge $m \to n$, then, when considering whether to add node m to A_0, this edge should not be counted towards the out-degree of m. This heuristic is applicable to all cascade models,

and is denoted here as the *Single discount* heuristic. In our experiments, often we found that *Single discount* will only minimally improve over *High-degree*, in which case we compare only against *Single discount*. Many other heuristics than the ones described above are known. Among those without optimality guarantees, Jiang et al. [7] tests Simulated Annealing under Independent Cascade, and finds that it has a complexity advantage over greedy heuristics, and can also find narrowly better solutions.

Papers [1,6,7] evaluate the heuristics above comparatively on a small number of large social networks, and generally find that, for these datasets, the *General greedy* algorithm and its improved variants do find better solutions than the two degree-based heuristics, if in cases only by a small percentage. To concretely illustrate the fact that degree-based heuristics can underapproximate significantly, Figure 3 gives two examples of small networks on which *High degree* is not effective. For the example on the left, *High degree* with $k = 2$ may select as optimal seeds the set $\{c, d\}$, from among other options of equal out-degrees. With Weighted Cascade, $\sigma(\{c, d\}) = 4$, while a better solution is the set $\{a, b\}$, with $\sigma(\{a, b\}) = 10$. On the right, under Independent Cascade with $p = 50\%$, the best single seed is $\{a\}$, with $\sigma(\{a\}) = 8.5$, rather than node b, which has the same degree, with $\sigma(\{b\}) = 6$.

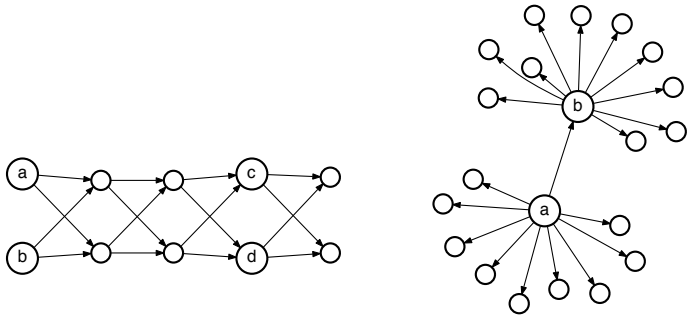


Fig. 3. Examples of networks executing cascade propagation models on which degree-based heuristics are ineffective. Left: a linear graph of nearly uniform node degrees, where *High degree* may choose $\{c, d\}$ as optimal seeds rather than the (here, optimal) $\{a, b\}$, under Weighted Cascade. Right: an extended star topology where a and b have equal degrees, but the seed set $\{a\}$ has higher influence, under Independent Cascade.

However, it is these degree heuristics without approximation guarantees which have the lowest computational complexity, rather than *General greedy*. Given a graph $G = (V, E)$, the problem size k, and number R of simulation repetitions to evaluate $\sigma(A_0)$, *High degree* has complexity $O(|V| \cdot \log(k))$ (with a min-heap implementation), while *General greedy* is $O(k \cdot |V| \cdot |E| \cdot R)$ [6]. In other words, for graphs that are large or dense, the *General-greedy* technique can be entirely infeasible.

3 Methodology: a Simple Genetic Algorithm

The application of Genetic Algorithms to the influence-maximization problem is straightforward: a candidate solution generated by the GA is encoded as a fixed-size sequence A_0 of k seed node (integer-valued) identifiers. For each solution A_0, we evaluate the fitness $\sigma(A_0)$ by running the Cascade Algorithm shown in Algorithm 1 (either according to the Independent Cascade or to the Weighted Cascade model). As the algorithm is stochastic, we run 100 simulation repetitions[4]. We then average, over the 100 repetitions, the size of A returned by the Cascade Algorithm. This average value is finally assigned to each candidate solution as its fitness.

The GA is configured to use 1-point crossover and a random mutation operator. The latter resets each node, with a given probability, to one of the possible node identifiers in G. Both genetic operators (crossover and mutation) are applied with probability one. Selection is performed using fixed-size tournament, with generational replacement and elitism (i.e., at each generation n_e best solutions in the population – the "elites" – are kept without being mutated). The complete list of parameters of the GA is reported in Table 1. We obtained these parameters through experimentation with different values; later in Section 4.3, we present the results of varying the population size, the number of elites, the tournament size, and the mutation rate.

Table 1. Parameters of the Genetic Algorithm

Parameter	**Value**
No. generations	$100 \times k$
No. elites	2 (for $k = \{10, 20\}$), 4 (for $k = \{30, 40, 50\}$)
Population size	100
Mutation rate (per node)	0.1
Crossover rate	1.0
Tournament size	5

The proposed parameters were empirically chosen by running preliminary experiments (see Section 4.3 on parameter analysis for further details). Overall, we observed that the performance of the GA is quite robust with different parameter settings, although we noted that some parameters (especially tournament size, mutation rate, and population size) can sensibly affect, as expected, the diversity

[4] This number of repetitions was chosen as a practical compromise between the confidence interval that it affords, and the overall computational complexity of the Genetic Algorithm. With regards to the accuracy of the fitness estimation, 100 simulation repetitions give a 95% confidence interval for the average in the approximate range of $[3, 10]$ nodes influenced, for all our experiments. Increasing the number of repetitions to 10000 would give a 95% confidence interval for the average that is ≤ 1 nodes influenced in all experimental cases, but requires far longer runtimes.

of the solutions found at the end of the evolutionary process, as well as the optimization results. As for the number of elites, we found that maintaining a slightly higher number of unmutated individuals (4 instead of 2) is especially beneficial for larger values of k. Indeed, as the search space dimensionality increases, keeping a larger group of diverse elites seems to prevent premature convergence and promote population diversity, especially in the later stages of evolution.

4 Experimental results and analysis

We test the proposed GA-based influence maximization algorithm on two large social network datasets, available from the SNAP repository [3]. A brief description of the datasets is reported in Table 2. It is to note that we chose networks with underlying graphs of strikingly different size and structure. The Wiki graph is relatively small at 7115 nodes, and very variate in terms of node degrees, with a large maximum out-degree. On the other hand, the Amazon graph (with over a quarter of a million nodes) is orders of magnitude larger, and has a flat degree landscape, with a maximum and median out-degree of 5.

Table 2. Large social networks from the Stanford large network dataset (SNAP [3]). The names in brackets indicate the dataset names used on the SNAP repository.

	Social network	**Graph type and size**	**Node out-degrees**
Wiki (wiki-Vote)	who-voted-whom in Wikipedia user elections (data collected on Jan 3 2008)	directed, 7115 nodes, 103689 edges	min 0, max 893, avg 14.57, stddev 42.28, median 2
Amazon (amazon0302)	Amazon product co-purchasing network (data collected on March 2 2003); if i is frequently co-purchased with product j, the graph has an edge $i \to j$	directed, 262111 nodes, 1234877 edges	min 0, max 5, avg 4.71, stddev 0.95, median 5

For each dataset, we consider the Independent and Weighted Cascade model with values of k (the size of the seed node identifiers) ranging in $\{10, 20, 30, 40, 50\}$. In the Independent Cascade model, we fix $p = 1\%$; this low probability is realistic for practical social networks. The combination between the Amazon dataset and the Independent Cascade model is an exception, in that the low probability p, together with the consistently very low node degrees in the Amazon graph, yield a "flat" fitness landscape. Therefore, we do not present the results of this combination.

In order to evaluate the influence of the initial population and the robustness of the algorithm, we execute the GA three times on each experimental condition, with different random seeds. All the experiments are implemented in Python, by using the Genetic Algorithm provided by the Python package `inspyred` [8] (configured as in Section 3). Experiments are run on two Linux machines: a Ubuntu 14.04 with 64 AMD Opteron 2.3 GHz cores and 256 GB RAM and a Ubuntu 12.04 with 32 Intel Xeon 2.0 GHz cores and 128 GB RAM. Computations are parallelized by running in multiple threads the evaluation of the candidate solutions generated by the GA.

A summary of the numerical results is shown in Figure 4 for the Wiki and Amazon datasets. In the figures, we compare the highest influence (i.e., the number of active nodes) obtained by the GA with the results obtained by random sampling and with the heuristics. All algorithms evaluate the fitness of a seed set A_0 in exactly the same way, i.e., by simulating the propagation model, starting with A_0, 100 times, and reporting the mean number of influenced nodes as fitness. The figures report this mean value together with a measure of the confidence in the mean: this is the 95% confidence interval, in the case of the algorithms which output a single solution (i.e., the degree heuristics *High degree* and *Single discount*, the greedy heuristic *General greedy*, and the Genetic algorithm), and the standard deviation in the case of random sampling, which samples 100 random values for A_0, each evaluated in 100 simulation repetitions. For the Wiki dataset, in the cases of both propagation protocols, the two degree-based heuristics, *High degree* and *Single discount*, yield results that are largely indistinguishable; because of this, we present only one in the figures, for clarity.

We also report, in Figure 5, examples of the generational trends on some of the test cases. The evolutionary trends show on each test case little variation among the various repetitions. Additionally, the convergence rate of the GA is quite high, especially on the Wiki dataset.

4.1 Comparative influence results

The fact that the random sampling of seed nodes yields solutions of low network influence is expected, as is the fact that the computationally intensive *General greedy* heuristic, known to fulfill an approximation guarantee (as motivated in Section 2.3) computes significantly better solutions.

On the other hand, the performance of the degree-based heuristics is surprising: on the Wiki dataset, the degree-based heuristics find better solution than *General greedy* for the larger k values, while on the Amazon dataset their performance is outdone by all other algorithms, including random sampling. This fact can be explained by the different structure of the underlying graphs: essentially, the Amazon graph conforms to the general idea shown in the example from Figure 3 (left), where we have shown that degree-based heuristics can be severely suboptimal. Overall, the degree heuristics have undependable performance.

Finally, the Genetic algorithm, although stochastic and not of an exhaustive nature (as *General greedy*), is consistently matching the *General greedy* heuristic;

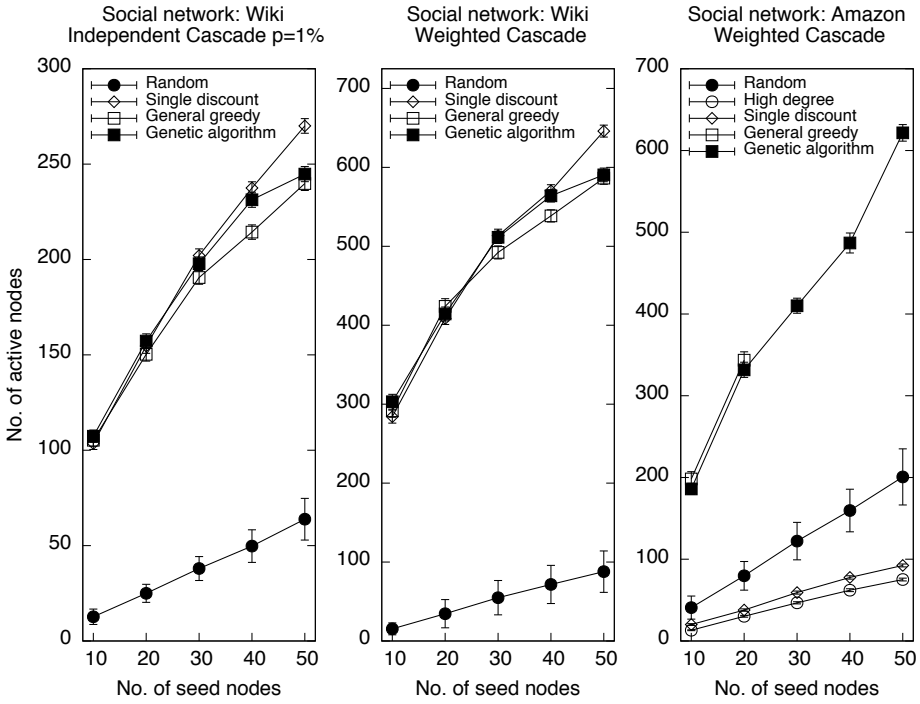


Fig. 4. Comparison of the numerical results: Wiki dataset, Independent Cascade (left); Wiki dataset, Weighted Cascade (middle); Amazon dataset, Weighted Cascade (right).

for some data points and the smaller Wiki dataset, the Genetic algorithm outdoes the greedy heuristic.

4.2 Runtimes

In Figure 6 we report the average runtimes, in core hours, of the Genetic Algorithm and General greedy heuristic over the entire set of test scenarios. We omit, for brevity, the runtimes of the other heuristics as they are, in comparison, orders of magnitude smaller (< 1 core hour).

We observe that, for the Amazon dataset, the Genetic Algorithm has a runtime that is approximately half as big as the runtime of the General greedy heuristic. On the other hand, on the Wiki network the GA is more computationally expensive (up to 30%) than the greedy heuristic, although for larger values of k the runtimes of the two algorithms are comparable, suggesting that on this particular dataset the GA is more efficient for larger k.

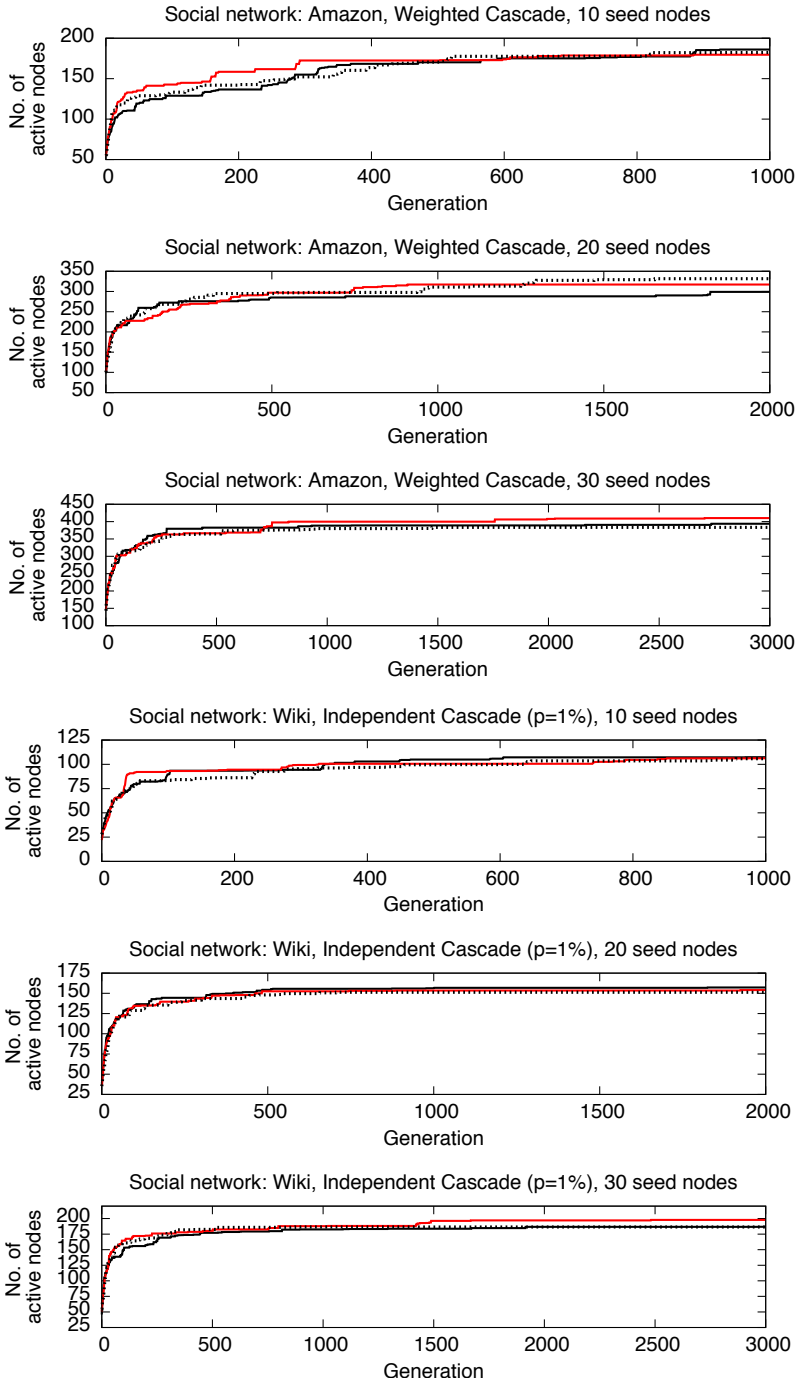


Fig. 5. Examples of generational trends observed in the experiments with the Genetic Algorithm. In each case, we report the trends of three repetitions of the GA.

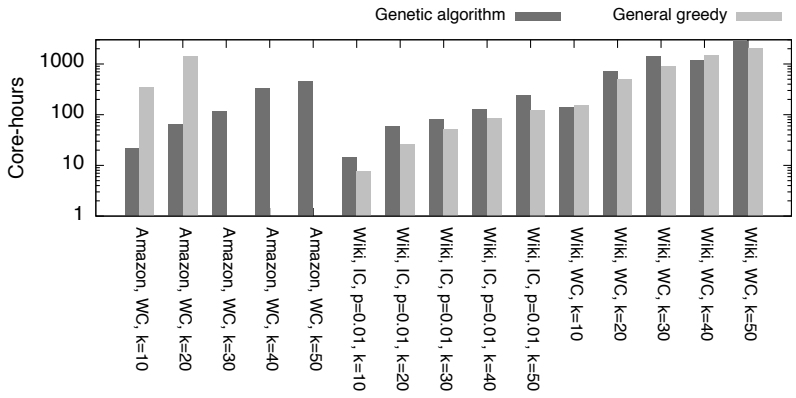


Fig. 6. Average runtimes, in core hours, of Genetic Algorithm and General greedy heuristic. Due to the limited computational resources available, we did not run the General greedy heuristic on the Amazon dataset for $k = \{30, 40, 50\}$.

4.3 Parameter analysis

We conclude our analysis by reporting some empirical observations we made in our experiments on the effect of the parameters of the GA on the optimization results. Due to limited computational resources, we performed only a preliminary qualitative analysis on a limited set of parameter configurations, to gain some general insight on the parametrization.

- **No. elites**: As reported in the previous section, elitism is beneficial as far as a small percentage ($2\% - 4\%$) of the best individuals in the population is maintained without undergoing mutation. Furthermore, for larger values of k there is a positive effect of higher numbers of elites on the diversity of individuals (particularly in the later stages of the optimization process).
- **Mutation rate**: We observed that smaller mutation rates (0.1) are overall beneficial in terms of optimization results, especially during the first stages of evolution. The other value we tested, 0.2, seems to produce disruptive mutations thus causing slower convergence. As shown in Figure 7 (top) for the case $\langle$Amazon, Weighted Cascade, $k = 10\rangle$, this trend appears in both cases of tournament sizes 2 and 5.
- **Tournament size**: The effect of tournament size shows mostly in two aspects: one one hand, increased tournament sizes increase the selection pressure [9], thus resulting in a higher convergence rate, see Figure 7 (top). Also, tournaments of 5 individuals allowed us to find a more diverse set of solutions compared to binary tournaments, especially for larger values of k.
- **Population size**: We tested two values of population size, namely 50 and 100. Overall, we noted that bigger populations lead to better optimization results, as shown in Figure 7 (bottom) for the case $\langle$Amazon, Weighted

Cascade, $k = 10\rangle$. The intuitive explanation is that larger populations allow a higher diversity level, thus improving the exploration of the search space.

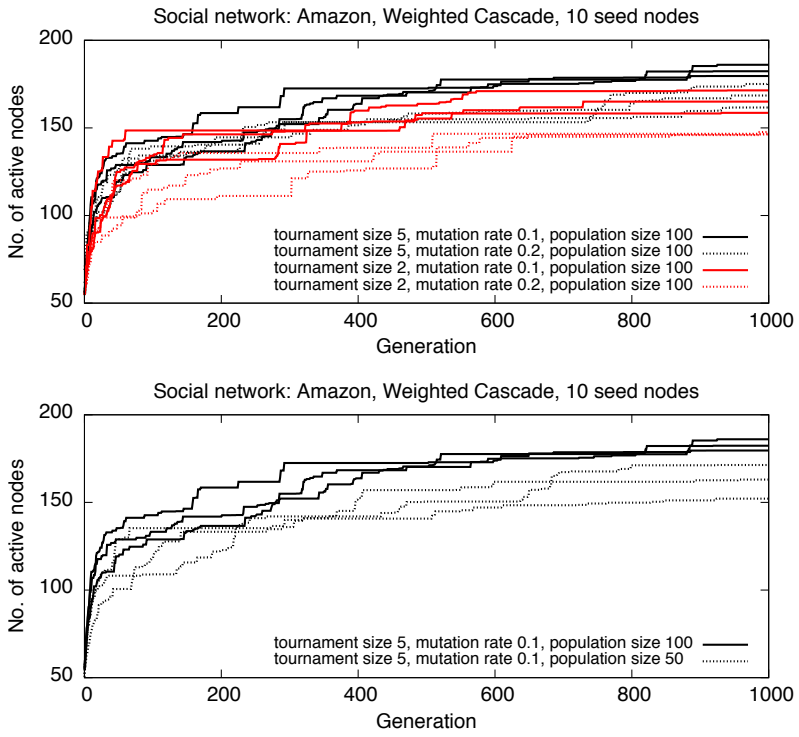


Fig. 7. Influence of the GA parameters on the optimization results: tournament size and mutation rate (top) and population size (bottom). For each parameter configuration, three repetitions of the GA are shown. The others parameters are kept constant, see Table 1 for details.

5 Conclusions

In this paper we made a first attempt to tackle the influence maximization problem in social networks by using Genetic algorithms. We performed an experimental campaign on two datasets taken from real-world applications, comparing the results of the GA against known heuristics based on either graph theory or a greedy, incremental exhaustive search. In short, our experiments showed that in each test case the performance of the GA is at least comparable to that of the best tested heuristic. However, as heuristics typically rely on specific network topology features, we observed a dramatic variation in their performance

from one case to the other. On the other hand, since the GA is agnostic w.r.t. the properties of the networks, its results were more consistent across the whole range of test cases. In addition to that, we found, quite surprisingly, that on some test cases the GA was computationally cheaper than the greedy heuristic.

To the best of our knowledge, our study represents one of the very first examples of application of computational intelligence methods to influence maximization in social networks, whereas a number of heuristics have been designed and tested on this problem. Overall, our experiments revealed that Genetic algorithms are a viable tool to solve the influence maximization problem, especially in absence of any prior knowledge about the network topology and the characteristics of the graph underlying the social network. Other related studies have also shown that bio-inspired algorithms are suitable to compute solutions to computationally hard problems from the area of complex systems and networks in polynomial time: [10,11] applied natural selection as a means to stress-test wireless sensor networks and find vulnerable topologies; [12,13] optimized the path of strong attackers in large-scale mobile, wireless, urban delay-tolerant networks, and uncovered potential vulnerabilities in a message-routing protocol for such networks.

Considering the high relevance of social networks in all modern applications, we believe it is worth to further investigate the application of computational intelligence to this domain. In future research, we plan to analyze more thoroughly the influence the GA parametrization on this specific problem. Furthermore, we deem promising the possibility to combine GAs with graph-based heuristics, in order to create domain-specific memetic algorithms. Finally, it will be interesting to extend the experimental setup to other datasets.

References

1. Kempe, D., Kleinberg, J., Éva Tardos: Maximizing the spread of influence through a social network. Theory of Computing **11**(4) (2015) 105–147
2. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In Fensel, D., Sycara, K., Mylopoulos, J., eds.: The Semantic Web - ISWC 2003. Volume 2870 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 351–368
3. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data` (October 2015)
4. Goldenberg, J., Libai, B., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters **12**(3) (2001) 211–223
5. Wang, C., Chen, W., Wang, Y.: Scalable influence maximization for independent cascade model in large-scale social networks. Data Mining and Knowledge Discovery **25**(3) (2012) 545–576
6. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '09, New York, NY, USA, ACM (2009) 199–208
7. Jiang, Q., Song, G., Cong, G., Wang, Y., Si, W., Xie, K.: Simulated annealing based influence maximization in social networks. In Burgard, W., Roth, D., eds.: AAAI, AAAI Press (2011)

8. Garret, A.L.: Inspyred: A framework for creating bio-inspired computational intelligence algorithms in Python. `https://pypi.python.org/pypi/inspyred` (October 2015)
9. Miller, B.L., Goldberg, D.E.: Genetic algorithms, tournament selection, and the effects of noise. Complex Systems **9** (1995) 193–212
10. Bucur, D., Iacca, G., Squillero, G., Tonda, A.: The impact of topology on energy consumption for collection tree protocols: An experimental assessment through evolutionary computation. Applied Soft Computing **16** (2014) 210–222
11. Bucur, D., Iacca, G., de Boer, P.T.: Characterizing topological bottlenecks for data delivery in CTP using simulation-based stress testing with natural selection. Ad Hoc Networks **30** (2015) 22–45
12. Bucur, D., Iacca, G., Squillero, G., Tonda, A.: Black holes and revelations: Using evolutionary algorithms to uncover vulnerabilities in disruption-tolerant networks. In Mora, A.M., Squillero, G., eds.: Applications of Evolutionary Computation. Volume 9028 of Lecture Notes in Computer Science. Springer International Publishing (2015) 29–41
13. Bucur, D., Iacca, G., Gaudesi, M., Squillero, G., Tonda, A.: Optimizing groups of colluding strong attackers in mobile urban communication networks with evolutionary algorithms. Applied Soft Computing **40** (2016) 416 – 426