Solving Influence Maximization Problem Under Deterministic Linear Threshold Model Using Metaheuristic Optimization

A. Andreev*, K. Chukharev*, S. Kochemazov*, A. Semenov*,
* ITMO University, St. Petersburg, Russia
veinamond@gmail.com

Abstract*—In the paper we consider the discrete variant of the well-known Influence Maximization Problem (IMP). Given some influence model, it consists in finding a so-called seed set of influential users of fixed size, that maximizes the total spread of influence over the network. We limit our study to the influence model called Deterministic Linear Threshold Model (DLTM). It is well known that IMP under DLTM is computationally hard and there are no approximate algorithms for its solving with a constant approximation ratio if $P \neq NP$. Therefore, it makes sense to apply metaheuristic algorithms to this problem. In the present research we propose new algorithms for solving IMP under DLTM, which are based on a technique that combines evolutionary and genetic strategies for pseudo-Boolean optimization with a greedy algorithm which is used to find some initial approximation. We use the proposed strategy to solve another well-known combinatorial problem for networks called Target Set Selection (TSS). We propose to solve TSS as a sequence of IMPs with gradually decreasing of target set size. In the experimental part of the paper we demonstrate that our new strategy outperforms the previous ways to solve TSS, yielding smaller target sets of good quality.

***Keywords*—Networks, Influence Maximization, Target Set Selection, Evolutionary Algorithms,(1+1)-EA, Metaheuristics**

I. Introduction

Graphs and networks are natural models of communities and one can use them to construct various types of functions describing dynamical processes in collectives. The processes of information dissemination in network have been studied in an enormous body of research, so mentioning even the key papers on this topic will take too much space. Thus in this context, we would like to cite the reviews [1], [2]. In many natural cases we can meet with the following problem: to set up some (relatively small) number of the so-called *active* agents in a network in such a way that after some period of time the majority of other agents in the network will became active too. One can consider the corresponding process as some kind of *influence spread* in a network: some agents can influence others more or less depending on the considered graph structure. Thus, using different ways of setting active agents at an initial moment of time we will see different outcomes of influence spread: some ways will result in a quicker activation process than the others. In this context, we can consider a problem of combinatorial optimization: to find such an activation set which has relatively small size and simultaneously activates a relatively large portion of all agents in the considered network. The corresponding questions are related to the so-called *Influence Maximization Problem (IMP)* [3].

In our paper we consider IMP and another combinatorial problem which is very close to IMP in a sense: we mean the so-called *Target Set Selection* (TSS) problem. As it is implied from results of [3], IMP and TSS are NP-hard combinatorial problems but they are very important for the study of the properties of real world big networks, and thus the following problem is relevant: to develop computational algorithms which solve IMP and TSS for large networks. Further we propose some new algorithms for these purposes. In our paper we deal with the *Deterministic Linear Threshold Model* (DLTM) for information dissemination [3], [4].

The main contributions of our paper are as follows. I) We propose a new kind of the (1+1) evolutionary strategy which makes it possible to optimize an arbitrary pseudo-Boolean function on a set of binary words of some length and fixed Hamming weight. II) Based on the proposed algorithm we develop a new iterative procedure for solving TSS as the sequence of local IMP problems. III) We apply the developed algorithms to some benchmarks from real world and demonstrate that the algorithms proposed in this paper solve TSS more efficiently than the known ones.

II. Preliminaries

Let us represent the collective as a directed labeled graph $G = (V, A, L)$, in which V is the set of vertices, also called *agents*, A is the set of arcs, and $L = L_V \cup L_A$ is the set of symbols called labels, in which L_V contains the labels associated with vertices, and L_A – the labels associated with arcs. Hereinafter we will refer to graph G as to *network*. In the present paper we assume that the information spreads within the network in discrete time moments $t \in \{0, 1, 2, \ldots\}$. We are particularly interested in the activation processes in the network: it is convenient to split the agents into active and inactive, and denote the corresponding states as 1 and 0, respectively. Assume, that at the initial time moment only some set T, $T \subseteq V$ of vertices are active, and they remain active indefinitely.

This work was supported by the Ministry of Science and Higher Education of Russian Federation, research project no. 075-03-2020-139/2 (goszadanie no. 2019-1339).

The vertices from $V\backslash T$ are inactive at moment $t = 0$, but can become active at the following time moments due to the influence of their neighbors. Following [5], we will refer to the agents from T as to *instigators*. The problem that is of interest here is to choose the set T of as small size as possible so that a portion exceeding some threshold of vertices from V will be activated by it. This formulation corresponds to the well-known Target Set Selection problem (TSS), and the set T is called a *target set* in this context. A similar problem to TSS is the Influence Maximization Problem (IMP), which implies the search for the best target set of a fixed size. IMP and a number of related discrete optimization problems were analyzed in the seminal paper [3]. Various theoretical properties of TSS have been later studied in, e.g. [6], [7], and the computational algorithms for its solving proposed in [4], [8], [9], etc.

The laws that govern the activation of network G by instigators from T can be defined by various means. In the present paper we will work with the Deterministic Linear Threshold Model (DLTM) [3], [4]. In particular, we assume that the state of an arbitrary vertex $v \in V\backslash T$ at time moment $t+1$, $t \geq 0$ is defined as follows:

$$a_v(t+1) = \begin{cases} 1, & \sum_{u\in U_v} a_u(t) \times w_{(u,v)} \geq \theta_v \\ a_v(t), & \text{otherwise} \end{cases} \quad (1)$$

In the equation (1) by $a_v(t)$ we denote the state of a vertex $v \in V$ at time moment t, expressed by the symbol from $\{0, 1\}$. U_v denotes the neighborhood of vertex v – the set of all vertices $u \in V$: $(u, v) \in A$, i.e. such that there is an arc from u to v. Next, $w_{(u,v)}$ is a rational number called the *weight* of an arc (u, v), and θ_v is a positive rational number called the *activation threshold* or simply *threshold* of vertex v. Thus, in the networks that we consider in the present work the activation is performed in accordance with the model first described in [5].

Let us formally define TSS as follows: for a given number $R \in \{1, \ldots, n\}$, $n = |V|$, to find TS T of the smallest size which activates at least R vertices in a network. By IMP in accordance with [3] we will consider the following problem: for a given $k \in \{1, \ldots, n\}$ to find a target set T of size k, which activates maximal number of vertices in G. As it follows from the results of [3], IMP is NP-hard. Moreover, for IMP under DLTM there can exist no approximate algorithm with a constant ratio factor if $P \neq NP$.

In [9] TSS was reformulated in form of a pseudo-Boolean optimization problem with an effectively computed fitness function. For this purpose, the authors associated with G a *Discrete Dynamic System* (DDS), the state of which at an arbitrary time moment $t \in \{0, 1, \ldots\}$ is described by a Boolean vector $W(t) = (w_1(t), \ldots, w_n(t))$, the coordinates of which correspond to the states of network vertices at time moment t. The transitions between states in accordance with (1) define the function of the following kind:

$$F_G : \{0, 1\}^n \rightarrow \{0, 1\}^n$$

where by $\{0, 1\}^n$ we denote the Boolean hypercube of dimension n, i.e. the set formed by all possible binary vectors of length n. The function F_G can also be defined by a graph, which is usually referred to as a *State Transition Graph* (STG). The sequences of states in STG can form cycles: when for some time moments $t_1, t_2 : t_1 < t_2$: $W(t_2) = W(t_1)$. The smallest value $t_2 - t_1$, for which it holds that $W(t_2) = W(t_1)$, is called the *length of cycle*. A cycle of length 1 is called a *fixed point*.

It is easy to see that DDS defined by DLTM for the initial conditions corresponding to TSS does not have cycles of length > 1 and it reaches the fixed point in at most n time moments. This fact means that one can associate with G the pseudo-Boolean function defined as follows.

$$\Phi_G : \{0, 1\}^n \rightarrow \{0, 1, \ldots, n\} \quad (2)$$

The argument of Φ_G is the Boolean vector $\alpha \in \{0, 1\}^n$, which defines the set T: the ones in α correspond to instigators that are in T at time moment $t = 0$. The remaining vertices are inactive at $t = 0$ and correspond to zeros in α. The value $\Phi_G(\alpha)$ is equal to the number of active vertices at the moment when DDS specified by network G reaches a fixed point starting from the initial point defined by vector α. Then TSS can be viewed as the problem of finding α (specifying T) with the minimum Hamming weight under the condition that $\Phi_G(\alpha) \geq R$.

III. TSS and IMP as Metaheuristic Optimization Problems

As we noted above, a number of papers employed different discrete optimization algorithms for solving TSS under DLTM. For example, in [8] essentially was considered the simplest case of IMP when the weights of all arcs are equal to 1. For this class of networks that paper employed the exact algorithms for finding target sets of fixed size with the help of state-of-the-art solvers for the Boolean satisfiability Problem (SAT).

In [4] there was proposed a greedy algorithm for this problem. Later, [10] described an improvement of the greedy heuristic from [4]. In [9] it was proposed to use the evolutionary algorithm (1+1)-EA to solve TSS, as well as a hybrid algorithm that combines (1+1)-EA with the greedy heuristic from [4]. In this section we describe a novel variant of the (1+1)-EA algorithm, that is aimed at solving IMP, and then describe the general strategy for solving TSS that includes the solving of IMP for vectors of fixed weight as a part of the strategy.

A. A Novel Variant of (1+1)-EA for Solving IMP

Remind, that the classical (1+1)-EA [11] is the basic evolutionary algorithm, which despite its simplicity has a number of nontrivial mathematical properties [12]. (1+1)-EA is very ineffective in the worst case scenario, but is surprisingly effective in practice. The main reasons for its effectiveness lie in the basic mutation algorithm and in the fact that on average (1+1)-EA behaves just like

Algorithm 1: The algorithm for computing $\mu_k^n(\cdot)$

```
Data: $\alpha$ is the input vector from $\{0,1\}_k^n$
Result: The result of $\mu_k^n(\alpha)$
// Split $\alpha$ into vectors of 1s and 0s, $|\alpha^1| = k$, $|\alpha^0| = n - k$
1  $\alpha^1, \alpha^0 \leftarrow$ Split($\alpha$);
2  $l = \min\{|\alpha^1|, |\alpha^0|\}$;
3  $\alpha' \in \{\alpha^1, \alpha^0\} : |\alpha'| = l$;
4  $\alpha'' = \{\alpha^0, \alpha^1\} \setminus \{\alpha'\}$;
5  $\widetilde{\alpha'} = \alpha'$;
6  $r = 0$;
7  for $i = 1, \ldots, l$ do
8      if Bernoulli_Trial ($1/l$) = True then
9          $\widetilde{\alpha'}[i] = 1 - \widetilde{\alpha'}[i]$;
10         $r \leftarrow r + 1$;
// Choose_Without_Returns $(a,b)$ function chooses $a$ numbers from 1,...,$b$ in accordance with the scheme without returns
11 $S'' =$ Choose_Without_Returns($r, n - l$);
// Flip $(a,\beta)$ replaces the values $\beta_i$ in vector $\beta$ with coordinates from $a$ by $1 - \beta_i$, e.g. flips them
12 $\widetilde{\alpha''}$=Flip ($S''$,$\alpha''$);
13 $\widetilde{\alpha}$ = Combine_vectors ($\widetilde{\alpha'}$,$\widetilde{\alpha''}$);
// It is clear that the Hamming weight of $\widetilde{\alpha}$ is $k$
14 return $\widetilde{\alpha}$
```

Algorithm 2: The proposed variant of (1+1)-EA

```
Data: $f : \{0,1\}^n \to \mathbb{R}$ is the pseudo-Boolean function to maximize, $k$ is the fixed Hamming weight, $N$ is the limit on the number of mutations
Result: The Best Known Value $f(\alpha^*)$ and the corresponding point $\alpha^*$, $|\alpha^*| = k$
// Find an initial point
1  $\alpha = 0^n$;
2  $S$ = Choose_Without_Returns ($k$,$n$);
3  $\alpha \leftarrow$ Flip($S, \alpha$);
4  Best_Known_Value = $f(\alpha)$;
5  while Number_of_mutations $< N$ do
6      $\widetilde{\alpha} = \mu_k^n(\alpha)$;
7      if $f(\widetilde{(\alpha)}) >$ Best_Known_Value then
8          $\alpha = \widetilde{\alpha}$;
9          Best_Known_Value = $f(\widetilde{(\alpha)})$;
10 return $f(\alpha)$, $\alpha$
```

Hill Climbing algorithm [13], meanwhile it still has the non-zero probability to transition from an arbitrary point $\alpha \in \{0,1\}^n$ to point $\alpha^* \in \{0,1\}^n$, in which the considered function achieves the global extremum. In the original variant of (1+1)-EA, with an arbitrary vector $\alpha \in \{0,1\}^n$ the algorithm associates the sequence of n independent Bernoulli trials with success probability $p = 1/n$ at each trial. If the trial number $i, i \in \{1,\ldots,n\}$ was successful, then the bit α_i in vector α is flipped (changed to the complementary one).

Now, let us describe a new variant of (1+1)-EA, which we will use to solve IMP. Define by $\{0,1\}_k^n$ the set formed by all vectors in the hypercube $\{0,1\}^n$ that have the Hamming weight k. Let $f : \{0,1\}^n \to \{0,1\}^n$ be an arbitrary pseudo-Boolean function. We want to find the point $\alpha^* \in \{0,1\}_k^n$, in which f achieves the maximal value over $\{0,1\}_k^n$. Let us generate a random vector $\alpha_0 \in \{0,1\}_k^n$: for this purpose we can choose k coordinates of vector α_0 w.r.t. a scheme without returns [14], and designate to these coordinates the value 1, and fill the remaining $n-k$ coordinates with value 0. Now, let us outline in α two vectors α^1 and α^0, formed by ones and zeros, respectively. Hereinafter, by $|\alpha|$ denote the length of an arbitrary vector α. Then, $|\alpha^1| = k$, $|\alpha^0| = n-k$. Introduce the parameter $l = \min\{|\alpha^1|, |\alpha^0|\}$ and let $\alpha' \in \{\alpha^1, \alpha^0\} : |\alpha'| = l$. Then, $\alpha'' = \{\alpha^0, \alpha^1\} \setminus \{\alpha'\}$. Consider the sequence of independent Bernoulli trials of length l with success probability $p = 1/l$. If in this sequence the trial with number i is successful, then in the vector α' the bit with number i is flipped. Otherwise, we leave the i-th bit unchanged. Denote the result of such a mutation of α' as $\widetilde{\alpha}'$. Let r be the number of bits in α', which were flipped. Now, consider the vector α'': $|\alpha''| = n-l$. Let us choose among them r coordinates w.r.t. scheme without returns [14] and flip the corresponding bits. Denote the results of this transformation of vector α'' by $\widetilde{\alpha}''$, and denote the obtained vector of size n with Hamming weight k as $\widetilde{\alpha}$. We will also denote the introduced transformation of vector α as $\mu_k^n(\alpha) = \widetilde{\alpha}$. The pseudocode of the algorithm for computing $\mu_k^n(\alpha)$ is presented at Algorithm 1.

In essence, we replace the mutation operator in (1+1)-EA by μ_k^n and slightly modify the general outline of the algorithm to account for the fixed Hamming weights k. It is presented in Algorithm 2. We will further refer to proposed variant of (1+1)-EA as to *(1+1)-weighted EA* or shortly *(1+1)-WEA*.

Let us determine the basic properties of the μ_k^n operator.

Theorem 1. *The following properties hold:*

1) *For each $k \in \{0,\ldots,n-1\}$ and for any two vectors $\alpha, \widetilde{\alpha} \in \{0,1\}_k^n$ there exists the transition of the kind $\mu_k^n(\alpha) = \widetilde{\alpha}$;*
2) $E[d_H(\alpha, \widetilde{\alpha})] = 2$.

Proof sketch. Let us prove the first property. Without the loss of generality, assume, that $l = k$ and let $\alpha, \widetilde{\alpha}$ be two arbitrary vectors from $\{0,1\}_k^n$. For an arbitrary $i \in \{1,\ldots,n\}$ consider the coordinates α_i and $\widetilde{\alpha}_i$ of these two vectors. Note, that for every variant: $(\alpha_i, \widetilde{\alpha}_i) \in \{(0,0),(0,1),(1,0),(1,1)\}$ the transition $\alpha_i \to \widetilde{\alpha}_i$ is possible as the result of the application of the operator

Algorithm 3: The proposed strategy for solving TSS

```
Data: $\Phi_G(\cdot)$ is the function that computes influence spread, $N$ is the limit on the number of mutations
Result: The result of solving TSS
// Find an initial point via greedy heuristic
1  $\alpha_{gr}$ = Find_using_greedy_algorithm ($\Phi_G$);
2  $k = |\alpha_{gr}|$;
3  Best_Point = $\alpha_{gr}$;
4  Best_Known_Value = $\Phi(\alpha_{gr})$;
   // Use heuristic to remove one vertex
5  $\alpha$ = Remove_Vertex($\alpha_{gr}$);
6  $k \leftarrow k - 1$;
7  Number_of_mutations = 0;
8  while Number_of_mutations < $N$ do
9      $\widetilde{\alpha} = \mu_q^n(\alpha)$;
10     if $\Phi_G(\widetilde{\alpha})$ > Best_Known_Value then
11         $\alpha = \widetilde{\alpha}$;
12         Best_Known_Value = $\Phi_G(\widetilde{\alpha})$;
13         Best_Point = $\widetilde{\alpha}$;
14         $k \leftarrow k - 1$;
15         $\alpha$ = Remove_Vertex($\widetilde{\alpha}$);
16 return Best_Known_Value, Best_Point
```

μ_k^n to α. It means that for any $\alpha, \widetilde{\alpha} \in \{0,1\}_k^n$ the outcome $\mu_k^n(\alpha) = \widetilde{\alpha}$ is possible.

Now let us prove the second property. By $d_H(\alpha, \widetilde{\alpha})$ denote the Hamming distance between vectors α and $\widetilde{\alpha}$. Since the transition $\alpha \to \widetilde{\alpha}$ is the result of the probabilistic experiment then $d_H(\alpha, \widetilde{\alpha})$ is a random variable. Note, that $d_H(\alpha, \widetilde{\alpha}) = 2d_H\left(\alpha', \widetilde{\alpha}'\right)$, since every flipped bit in α' corresponds to exactly one flipped bit in α''. But the mutation strategy of vector α' is the classical (1+1) random mutation with success probability $p = 1/|\alpha'|$. It is also well known that in this case $d_H\left(\alpha', \widetilde{\alpha}'\right) = 1$, thus, taking into account the properties of the expected value we have the validity of the second part of the statement of the Theorem. Thus, Theorem 1 is proved. □

Note, that it makes sense to employ the described variant of (1+1)-EA only in the cases, when there exists TSS, the size of which is significantly smaller than n: indeed, for a fixed k, the number of words of weight k in $\{0,1\}^n$ grows as a polynomial with the increase of n due to the well known inequality for binomial coefficients: $\binom{n}{k} \leq \frac{n^k}{k!}$. Therefore, for small k (significantly smaller than $n/2$) the ponential search space over which the considered function is optimized is substantially smaller than $\{0,1\}^n$.

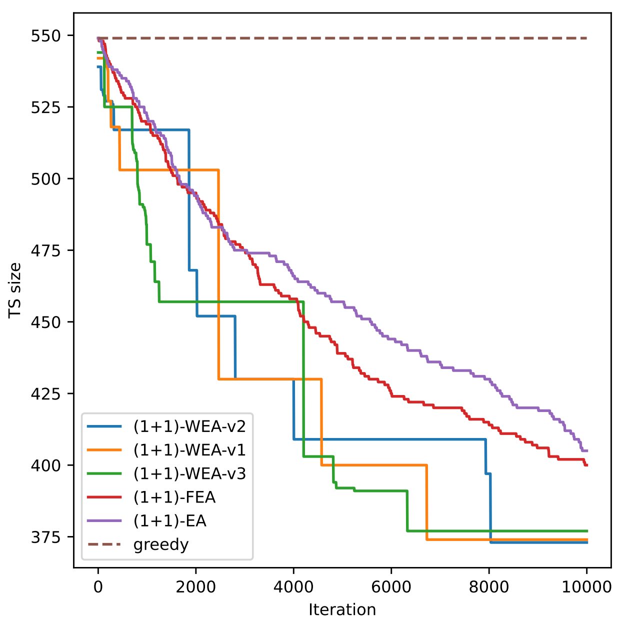

Fig. 1. Convergence plot for a single launch of Greedy algorithm (dashed) and combinations of Greedy with: (1+1)-EA (violet), (1+1)-FEA (red) and with three configurations of (1+1)-WEA (orange, blue and green), for 'p2p-Gnutella06_const_0.8(8717)' problem with constant thresholds and with limitation of 10^4 mutations.

B. The Strategy for Solving TSS Using the New Variant of (1+1)-EA

Below we describe the general strategy for solving TSS that we use in the computational experiments.

1) Consider the problem of finding an approximate solution of TSS for the considered network G.
2) Find the approximate solution of TSS – the vector $\alpha^\star$, using the algorithm from [4].
3) Let $\alpha^\star \in \{0,1\}^n$ be the solution found at the previous step and $wt(\alpha^\star) = k$ be the Hamming weight of $\alpha^\star$. We intend to transition to IMP over the set $\{0,1\}_{k-1}^n$. Let us construct some starting point β in $\{0,1\}_{k-1}^n$. For this purpose we consider the target set $T(\alpha^\star)$ specified by vector $\alpha^\star$ and remove from $T(\alpha^\star)$ a vertex v using some heuristic criterion. The remaining set of vertices corresponds to some vector $\beta \in \{0,1\}_{k-1}^n$ and the algorithm (1+1)-WEA starts its work from this vector. If there exists $\beta^\star \in \{0,1\}_{k-1}^n : \Phi_G(\beta^\star) \geq \Phi_G(\alpha^\star)$ (remind that $\Phi_G(\cdot)$ is a function of the kind (2)), then $\alpha^\star := \beta^\star$ (the point $\beta^\star$ becomes the current point), and we transition to IMP for $\{0,1\}_{k-2}^n$, and so on.
4) As we will see further, the effectiveness of solving IMP on $\{0,1\}_{k-1}^n$ crucially depends on the method that we use to choose a vertex for removing from $T(\alpha^\star)$. In the experiments we use for this purpose several natural heuristics more detail information about which will be presented below.

The pseudocode of the described strategy based on the use of (1+1)-WEA is presented at Algorithm 3.

TABLE I

EXPERIMENTAL RESULTS FOR LARGE NETWORKS. EVOLUTIONARY ALGORITHMS WERE ALLOWED TO PERFORM UP TO 10^4 ITERATIONS. THE RESULTS OF NONDETERMINISTIC ALGORITHMS ARE AVERAGED OVER 20 LAUNCHES AND ARE PRESENTED IN CURSIVE.

	greedy	(1+1)-EA	(1+1)-FEA	(1+1)-WEA-V1 $\beta = 3$	(1+1)-WEA-V2	(1+1)-WEA-V3
ca-GrQc_const_0.8 (5242)	*1635.00* *(1.92 s)*	*1547.35* *(155.72 s)*	*1544.40* *(142.13 s)*	***1526.85*** *(139.38 s)*	*1530.75* *(141.36 s)*	*1532.55* *(137.15 s)*
ca-GrQc_uniform_0.75-1 (5242)	*1522.00* *(1.96 s)*	*1429.65* *(152.87 s)*	*1426.00* *(141.09 s)*	*1408.30* *(138.47 s)*	***1407.25*** *(140.83 s)*	*1417.35* *(138.13 s)*
facebook_combined_const_0.8 (4039)	*1546.00* *(18.30 s)*	*1413.20* *(802.44 s)*	*1405.00* *(867.42 s)*	***1370.10*** *(854.30 s)*	*1378.50* *(852.75 s)*	*1425.05* *(836.13 s)*
facebook_combined_uniform_0.75-1 (4039)	*1360.00* *(17.66 s)*	*1216.00* *(750.71 s)*	*1203.65* *(797.93 s)*	*1206.95* *(807.50 s)*	*1204.65* *(796.08 s)*	***1221.00*** *(786.24 s)*
p2p-Gnutella06_const_0.8 (8717)	*549.00* *(14.81 s)*	*405.20* *(327.57 s)*	*397.00* *(289.51 s)*	*391.35* *(124.63 s)*	*388.20* *(157.81 s)*	***377.20*** *(175.14 s)*
p2p-Gnutella06_uniform_0.75-1 (8717)	*459.00* *(16.22 s)*	*336.65* *(325.13 s)*	*332.85* *(293.11 s)*	*325.60* *(121.62 s)*	*326.45* *(153.22 s)*	***308.65*** *(165.79 s)*
p2p-Gnutella08_const_0.8 (6301)	*300.00* *(7.67 s)*	*214.50* *(185.64 s)*	*212.80* *(162.24 s)*	*207.90* *(77.03 s)*	*212.40* *(89.04 s)*	***200.90*** *(102.12 s)*
p2p-Gnutella08_uniform_0.75-1 (6301)	*249.00* *(8.06 s)*	*172.80* *(187.41 s)*	*168.65* *(164.94 s)*	*171.85* *(73.39 s)*	*169.70* *(88.45 s)*	***158.20*** *(94.89 s)*
Wiki-Vote_const_0.8 (7115)	*3450.00* *(69.94 s)*	*3191.50* *(520.21 s)*	*3194.30* *(532.68 s)*	*3193.45* *(534.98 s)*	*3186.80* *(535.75 s)*	***3178.60*** *(543.65 s)*
Wiki-Vote_uniform_0.75-1 (7115)	*3391.00* *(72.03 s)*	***3143.20*** *(506.03 s)*	*3146.90* *(516.29 s)*	*3152.75* *(536.02 s)*	*3148.90* *(527.41 s)*	*3143.80* *(533.57 s)*

IV. COMPUTATIONAL EXPERIMENTS

In the computational experiments we worked with the fragments of real-world networks taken from the Stanford network repository [15]. Specifically, the follwing classes of networks were considered: 'facebook_combined'; 'wiki-vote'; 'p2p-Gnutella06'; 'p2p-Gnutella08'; 'ca-GrQc'. The original networks were undirected, thus each edge of the original graph was replaced by a pair of arcs with opposite directions.

Further, in Table I we add to the instance name the details about the thresholds generation scheme as well as the number of vertices. For each considered network we generated two different thresholds distributions. In both cases the threshold of an arbitrary vertex $v \in V$ was defined in accordance with the following formula: $\theta_v = \lceil \varepsilon \times (\sum_{u \in U_v} w_{(u,v)}) \rceil$. In the first case (equal thresholds for all vertices) we used $\varepsilon = 0.8$, while for the second one ε was chosen randomly for each vertex from $[\frac{3}{4}, 1]$. Thus, e.g. the notation 'wiki_vote_const_0.8_(7115)' means that we considered the 'wiki_vote' graph from [15] with 7115 vertices and constant thresholds for each vertex.

All experiments were carried out on the computing cluster node equipped with two 32-core Intel Xeon Gold 6242 CPUs and 1 TB RAM.

When solving TSS we were searching for a minimal TS that activates $\geq 75\%$ of all vertices in the network. Let $T(\alpha^*)$ be the target set found by the greedy algorithm from [4] and $T(\beta^*)$ be the target set found by the novel variant of (1+1)-EA ((1+1)-WEA), thus $T(\beta^*) \subset T(\alpha^*)$. Each time when searching for β^* we demand that the found $T(\beta^*)$ activates at least as many vertices as $T(\alpha^*)$. If such a $T(\beta^*)$ is found in some $\{0,1\}^n_k$, then we transition to searching for TS with the same properties in $\{0,1\}^n_{k-1}$. To move from $\{0,1\}^n_k \to \{0,1\}^n_{k-1}$ we remove from TS $T(\alpha^*) : \alpha^* \in \{0,1\}^n_k$ a single vertex in accordance with some heuristic. The resulting vector $\beta \in \{0,1\}^n_{k-1}$ becomes the initial point for (1+1)-WEA in the search space $\{0,1\}^n_{k-1}$.

As it can be seen from the results of experiments, see Figure 1 and Table I, for the (1+1)-WEA-based strategy the method of removing a vertex from the set $T(\alpha^\star)$ is critically important. For this purpose we considered three different heuristics. In accordance to the first one, which we denote as (1+1)-WEA_v1, the vertex with the smallest degree in G is removed from $T(\alpha^\star)$. In the modification (1+1)-WEA_v2 we remove the vertex with the smallest total sum of outgoing arcs. Finally, the modification (1+1)-WEA_v3 evaluates for an arbitrary vertex $v \in T(\alpha^\star)$ the value $\pi(v)$ called *activation potential* of v: for an arbitrary arc (v,u) with weight $w_{(v,u)}$ we compute $\pi(v) = \sum_{u:(v,u)\in A} \frac{w_{(v,u)}}{\theta(u)}$ and remove from $T(\alpha^\star)$ the vertex with the smallest $\pi(v)$.

In all experiments we used for (1+1)-WEA, classical (1+1)-EA and (1+1)-FEA from [16] the same limitation on the number of mutations, namely, each algorithm was allowed to perform up to 10^4 mutations.

Let us comment on Table I. As we mentioned above, in the experiments we used the networks from [15] modified to account for the need for directed edges, and with two schemes of thresholds' generation. In total we constructed 10 tests. For each of them we constructed the initial

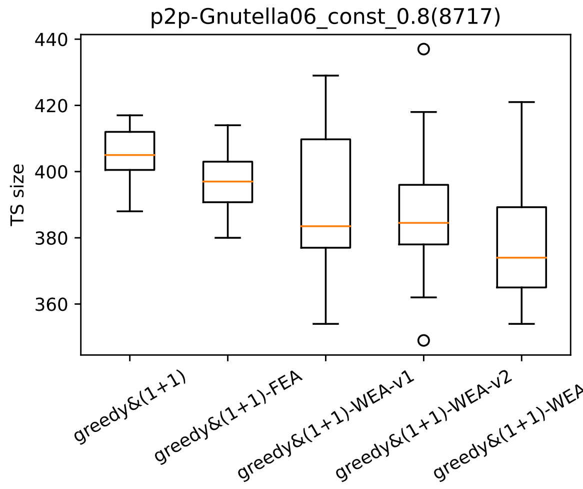

Fig. 2. Results of statistical testing for the instance p2p-Gnutella06_const_0.8.

approximation using the Greedy algorithm from [4], after which we employed the following algorithms: (1+1)-EA (classical (1+1)-EA, [11]), (1+1)-FEA (the variant of (1+1)-EA from [16]) and (1+1)-WEA (three variants of (1+1)-EA proposed in the present paper). For each network and each algorithm we considered 20 independent runs, and the table shows the average size for the TS found, and the average time in seconds.

At the Fig. 1 the convergence plot is presented for one run of all considered algorithms on the instance 'p2p-Gnutella06_const_0.8(8717)'. The difference in behavior of (1+1)-WEA and other forms of (1+1)-EA is worth noting. The ladder-like form of the convergence plot for (1+1)-WEA is a consequence of the gradual reduction of the weight of the current target set by 1.

To assess the statistical significance of our result, we conducted a non-parametric Wilcoxon signed-rank test on the samples between the standard greedy&(1+1)-EA and the best result of one of the three algorithms: greedy&(1+1)-WEA-V1, greedy&(1+1)-WEA-V2, greedy&(1+1)-WEA-V3. We considered the following statement as the null hypothesis: "*two populations have the same distribution*". Rejection of the null hypothesis also implies that the differences between the two samples are statistically significant. After conducting the Wilcoxon signed-rank test in 9 out of 10 tasks, we were able to reject the null hypothesis with a significance level of $\alpha = 0.05$. We also show the standard boxplots for one of the considered instances on Figure 2.

V. Conclusion

The main result of the present paper is the novel strategy for solving the Target Set Selection problem as a series of Influence Maximization Problems via a special evolutionary algorithm. In particular, we developed a variant of the (1+1)-evolutionary algorithm that optimizes an arbitrary pseudo-Boolean function over a subset of the Boolean hypercube, formed by the vectors of fixed weight. We demonstrate that the proposed algorithm when combined with the strategy in which we employ the greedy algorithm to construct the initial approximate solution yields substantially better results compared to the standard (1+1)-EA [11] or (1+1)-FEA from [16]. In the nearest future, we plan to implement the proposed algorithms for solving TSS and IMP on GPUs.

References

[1] M. E. J. Newman, "The structure and function of complex networks," *SIAM Rev.*, vol. 45, no. 2, pp. 167–256, 2003.
[2] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, "Critical phenomena in complex networks," *Rev. Mod. Phys.*, vol. 80, pp. 1275–1335, 2008.
[3] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*. ACM, 2003, pp. 137–146.
[4] A. Swaminathan, "An Algorithm for Influence Maximization and Target Set Selection for the Deterministic Linear Threshold Model," Master's thesis, Virginia Polytechnic Institute and State University, USA, 2014.
[5] M. Granovetter, "Threshold models of collective behavior," *American Journal of Sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.
[6] O. Ben-Zwi, D. Hermelin, D. Lokshtanov, and I. Newman, "Treewidth governs the complexity of target set selection," *Discret. Optim.*, vol. 8, no. 1, pp. 87–96, 2011.
[7] E. Ackerman, O. Ben-Zwi, and G. Wolfovitz, "Combinatorial model and bounds for target set selection," *Theor. Comput. Sci.*, vol. 411, no. 44-46, pp. 4017–4022, 2010.
[8] S. Kochemazov and A. Semenov, "Using synchronous boolean networks to model several phenomena of collective behavior," *PLOS ONE*, vol. 9, no. 12, pp. 1–28, 12 2014.
[9] M. Smirnov, S. Kochemazov, and A. A. Semenov, "The study of the target set selection problem under deterministic linear threshold model using evolutionary algorithms," in *MIPRO*. IEEE, 2023, pp. 1039–1044.
[10] F. Gursoy and D. Günneç, "Influence maximization in social networks under deterministic linear threshold model," *Knowl. Based Syst.*, vol. 161, pp. 111–123, 2018.
[11] H. Mühlenbein, "How genetic algorithms really work: Mutation and hillclimbing," in *PPSN*. Elsevier, 1992, pp. 15–26.
[12] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, no. 1-2, pp. 51–81, 2002.
[13] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.
[14] W. Feller, *An Introduction to probability theory and its applications*, 3rd ed. John Wiley & Sons, Inc., 1968, vol. 1.
[15] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.
[16] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *GECCO*, 2017, p. 777–784.