Variable Neighborhood Search for Weighted Total Domination Problem and Its Application in Social Network Information Spreading

Stefan Kapunac[a,*], Aleksandar Kartelj[a], Marko Djukanović[b]

[a] *Faculty of Mathematics, University of Belgrade, Serbia*
[b] *Faculty of Natural Sciences and Mathematics, University of Banja Luka, Bosnia and Herzegovina*

Abstract

We propose variable neighborhood search (VNS) as a solution to the weighted total domination problem (WTDP). A carefully designed fitness function allows for evaluation of both feasible and infeasible solutions, which further allows for a thorough search of the promising regions of the solution space. The method also utilizes two effective first-improvement local search procedures. The effectiveness of VNS is demonstrated on a wide range of benchmark sets compared to all three competing methods from the literature. For small-to-middle-sized instances (up to 100 nodes), VNS can obtain solutions that match the quality of optimal solutions in almost all cases (134 out of 135). For middle-to-large-sized instances, VNS can outperform all comparison algorithms in terms of solution quality, which is verified by statistical hypothesis tests. In addition, we present a potential application of the WTDP for boosting information spreading across social networks. Experiments confirmed that information spreading is accelerated when informed nodes (spreaders) are set to be solutions of the WTDP obtained by VNS.

Keywords: Domination problems, variable neighborhood search, SIR models, information spreading

1. Introduction

From the 1950s onward, the class of *dominating problems* [10, 1] have been particularly interesting both from theoretical and practical points of view. Dominating problems still keep this field highly attractive to many researchers. Domination in graphs has various applications such as error correcting codes, random testing, personnel shift planning, routing in wireless sensor and optical networks [49, 46], social networks [5], bioinformatics [28], allocating wavelengths to a broadcast connection [45], solving the five-queens problem [44], etc.

The basic representative of this class is *the minimum dominating set problem* (MDSP), which has been proved $\mathcal{NP}$–hard by Karp. Let us define graph $G = (V, E)$, where V is a set of nodes (vertices) and E is a set of edges describing a binary relation on the set $V \times V$, more formally $E = \{\{u, v\} \mid u, v \in V \wedge u \neq v\}$. The problem is to find a minimum cardinality set $S \subseteq V$ such that for each $v \in V$ at least one of the following two conditions is satisfied: 1) $v \in S$ or 2) there is at least one edge $e = \{v, u\} \in E$ such that $u \in S$. There are many

*Corresponding author
Email addresses: stefan_kapunac@matf.bg.ac.rs (Stefan Kapunac), kartelj@matf.bg.ac.rs (Aleksandar Kartelj), marko.djukanovic@pmf.unibl.org (Marko Djukanović)

related variants of this problem as well as generalizations therein, which have been studied over the last few decades. These include: the minimum set cover problem [8], the minimum k-dominating set problem [12], the minimum positive influence dominating set problem [5], the minimum total dominating set problem [27], the minimum capacitated dominating set problem [41], the weighted total domination problem, etc.

This paper focuses on solving the recently introduced *weighted total domination problem* (WTDP) [36].

1.1. Notation and problem definition

Before providing the WTDP definition, let us introduce the notation which will be used throughout the paper. For $v \in V$, by $N(v)$ we define the set of neighboring nodes (or neighborhood) of node v, i.e. $N(v) = \{u \mid \{u, v\} \in E \wedge u \neq v\}$. By $N[v]$ we define the closed set of neighboring nodes (or closed neighborhood) of node v, i.e. $N[v] = N(v) \cup \{v\}$. A set $S \subseteq V$ is called the *dominating* set in graph G iff for each $v \in V$, either $v \in S$ or $|N(v) \cap S| \geq 1$. Shortly, for each $v \in V$ it must hold that $|N[v] \cap S| \geq 1$. A dominating set $S \subseteq V$ is called a *total dominating* set iff for each $v \in V$ it must hold that $|N(v) \cap S| \geq 1$.

An instance of the WTDP is a weighted graph $G = (V, E, f_1, f_2)$, where $f_1 \colon V \mapsto \mathbb{R}^+$ is a node weight function, and $f_2 \colon E \mapsto \mathbb{R}^+$ an edge weight function on graph G. The aim of the WTDP is to find a set $S \subseteq V$ that fulfills the following two conditions:

1. S is the total dominating set of graph G;
2. S is the set with the minimum *objective* value calculated as:

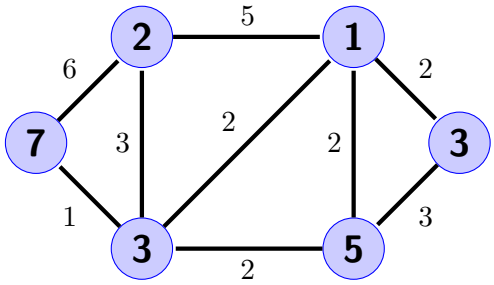
$$obj(S) = \sum_{v \in S} f_1(v) + \sum_{e=\{u,w\} \in E[S]} f_2(e) + \sum_{v \in V \setminus S} \min\{f_2(e = \{v, w\}) \mid w \in S\}, \qquad (1)$$

where $G[S] = (S, E[S])$ represents a sub-graph of G induced by S.

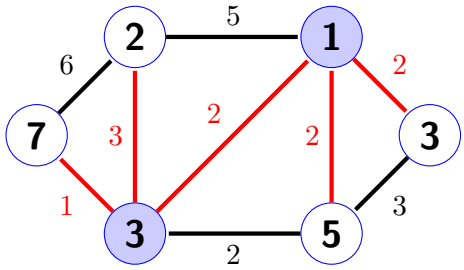
Following the notation in [2], the value of the first sum is called the *node selection cost*, the value of the second sum is called the *internal edge cost* whereas the value of the third sum of the objective function (1) is called the *external edge cost*.

WTDP is introduced in [36] as a generalization of the vertex-weight total domination problem, that uses weights on the node basis only. Further, the vertex-weight total domination problem itself is the generalization of the (unweighted) total domination problem TDP, which is a well-studied problem introduced in [11]. TDP is $\mathcal{NP}$–hard for general graphs, see [22]. For further details, we refer interested readers to a TDP survey in [25].

1.2. Exemplary problem instance



(a) Problem instance.
The numbers represent weights of nodes and edges.



(b) The optimal solution of the problem instance (a). The set S consists of the nodes filled in blue color. Red edges contribute to the second and third sum of the objective function. The objective function value is $(1 + 3) + 2 + (1 + 3 + 2 + 2) = 4 + 2 + 8 = 14$.
Best viewed in color.

2. Related work

MDSP has been intensively studied over the last decades. Ever since its introduction, one of the main research interests has been determining the upper and lower bounds of the domination number, i.e., the size of the optimal solution of MDSP (see [3, 4, 30, 39]). The bounds are established for general graphs, as well as some special graph classes such as bipartite, chordal graphs, or de Bruijn digraphs. The computational complexity of the problem from the theoretical point of view was studied in [9, 13, 33, 18, 15]. Concerning the approximation algorithms for MDSP, we refer the reader to [19]. Exact algorithms for MDSP are proposed in [16, 47]. The following heuristic algorithms are also proposed for dealing with large-sized instances: ant colony optimization approach [26], hybrid genetic algorithm [23], simulated annealing approach [24], and local search framework [7], which proved to be the state-of-the-art algorithm for huge graph instances.

As already mentioned, there are many variants of the MDSP which provide certain practical applications. We will just name a few: *the minimum capacitated dominating set problem* (MCDSP), *the minimum connected dominating set problem* (MCDSP), and the (minimum) *weighted total domination* (WTDP), which is in the focus of the present work. The current state-of-the-art algorithm for MCDSP is an ant-colony based approach [6], and for MCDSP it is the hybrid evolutionary algorithm [40].

There are several ILP models proposed for solving WTDP. Three models are proposed in Yuede et al. in [36]. Two more enhanced Mixed-Integer Programming models are proposed by Alvarez-Miranda and Sinnl in [2]. In addition, the same authors introduced the solution framework that utilizes valid inequalities (by exploiting the power of Benders' decomposition), starting heuristics and primal heuristics. Also, in the same paper, the authors proposed a genetic algorithm combined with a greedy randomized adaptive search procedure.

2.1. Our contributions

The current state-of-the-art for WTDP is the exact branch-and-cut (B&C) framework based on the two ILP models from [2], improved by a starting and primal heuristic within the search process. Performances of this framework are verified for small and middle-sized instances (the graphs up to 125 nodes), as the B&C framework solves almost all instances to optimality. Besides this exact approach, there is a general lack of non-exact (heuristic)

approaches in the literature which deals with large-sized instances. There are only two such approaches: a greedy randomized adaptive search procedure (GRASP) and a genetic algorithm (GA) hybridized with GRASP, in further text denoted as GRASP+GA. The latter was highly effective since the obtained solutions were able to match the quality of almost all (known) optimal solutions of the considered benchmark sets. However, the pros and cons of the B&C and GRASP+GA have not been thoroughly investigated since their effectiveness for large-sized instances has not been assessed. Moreover, more advanced and effective heuristic techniques have neither been developed nor compared to GRASP+GA.

The contributions of the present work are as follows.

- We introduce an efficient variable neighborhood search (VNS) approach to deal with WTDP. The approach utilizes a fitness function that allows the method to make successful decisions through both feasible and infeasible parts of the solution space. In addition, two effective local search procedures are proposed within the VNS scheme.
- In order to further investigate the benefits and limitations of the proposed approaches from the literature and our approach, we additionally generate large-sized instances with up to 1000 nodes. The performances of GRASP+GA, VNS, GRASP and ILP approaches are comprehensively evaluated in the experimental section.
- The quality of the proposed VNS is validated by the following facts:
  - VNS achieves optimal solutions for all small-sized instances;
  - it significantly outperforms GRASP+GA on medium-sized instances;
  - it significantly outperforms GRASP+GA on the newly introduced large-sized instances.
- We provide an innovative application of the WTDP solution for improving information spreading in social networks. We demonstrate that information spreading convergence is improved by informing individuals (nodes) in the social network (underlying graph) that are a part of the WTDP solution. This is supported by comparing two approaches for selecting initially informed individuals:
  1. when these are selected as a part of WTDP solution obtained by VNS;
  2. when the same number of individuals is selected randomly (baseline).

The rest of the paper is organized as follows. In Section 3 we propose the VNS approach and describe it in detail. Section 4 presents excessive experimental evaluation. Section 5 shows how the solution to WTDP is involved in information spreading across social networks. Section 6 provides conclusions and outlines directions for future work.

3. The proposed method

Variable neighborhood search (VNS) is a well-known metaheuristic proposed by Mladenović and Hansen [37]. The core idea of the algorithm is the systematic change of neighborhood structures with the purpose to escape from local optima. In the past two decades, VNS has proven to be a highly effective metaheuristic, successfully applied to various $\mathcal{NP}$-hard optimization problems (see [20, 21]).

Algorithm 1 gives an outline of the proposed VNS for WTDP.
VNS algorithm is generally comprised of two main parts: *shaking* and *local search*.

1. Shaking randomly selects a new candidate solution within the current neighborhood at each iteration. The goal of the shaking is to escape local optima. This is done by introducing a certain level of solution diversification through random movement in the solution space, with the level being controlled by the size of the current neighborhood. This means that a larger neighborhood induces higher diversification. More details on shaking will be provided in Section 3.3.
2. Local search (LS) has the opposite goal, as it focuses on solution search intensification. The candidate solution, obtained by performing shaking, is systematically improved by checking the local neighborhood around the candidate solution. We use a combination of two LS procedures, which are thoroughly explained in Section 3.4.

VNS main loop iterations (comprised of shaking and local search steps) are performed until termination criteria are met. In our case, these are: 1) the maximal number of iterations it_{max} has been reached and 2) the maximal execution time in seconds t_{max} has been reached. When either of these two criteria is met (logical *or*), the search process stops. Or conversely (as in pseudo-code), as long as neither of these two criteria are met (logical *and*), the search process is active.

Algorithm 1 VNS for the WTDP.

Input: $G = (V, E, f_1, f_2)$: problem instance; $\kappa_{\min}$: minimal neighborhood size; $\kappa_{\max}$: maximal neighborhood size; it_{max}: maximal number of iterations; t_{max}: maximal execution time in seconds.
$S \leftarrow$ `Random`()
$it \leftarrow 0$
$\kappa \leftarrow \kappa_{min}$
while $it < it_{max} \wedge t_{elapsed} < t_{max}$ **do**
    $S_{new} \leftarrow$ `Shaking`(S, κ, G)
    `LocalSearch1`(S_{new}, G)
    if S not improved recently $\wedge$ `LocalSearch2` not called recently **then**
        `LocalSearch2`(S_{new}, G)
    end if
    $RV \leftarrow \mathcal{U}(0, 1)$
    if $fit(S_{new}, G) < fit(S, G) \ \vee (fit(S_{new}, G) = fit(S, G) \ \wedge RV < 0.5)$ **then**
        $S \leftarrow S_{new}$
        $\kappa \leftarrow \kappa_{min}$
    else
        $\kappa \leftarrow \kappa + 1$
        if $\kappa \geq \kappa_{max}$ **then**
            $\kappa \leftarrow \kappa_{min}$
        end if
    end if
    $it \leftarrow it + 1$
end while
Output: S

In further subsections we describe all relevant aspects of the proposed VNS method: solution representation and initialization, fitness function, solution initialization, shaking and

moving, local search procedures and partial fitness function.

3.1. Solution representation and initialization

VNS encodes solution S naturally through a set data structure, also denoted as S in our pseudo-codes. Other aspects of the solution, namely, internal and external edges introduced in Section 1.1, are calculated based on S. Therefore, they are not true decision variables, but dependent ones.

The initial solution is constructed by choosing nodes randomly. Each node has the probability 0.2 to be included in the initial solution. This probability is set empirically.

3.2. Fitness function

Fitness function $fit(S)$ is based on objective function $obj(S)$ introduced in Equation (1). It is, however, more subtle since it allows for evaluation of both feasible and infeasible solutions. That way VNS solution search space is not restricted to feasible regions exclusively, which proves to be useful for the overall search process.

$$fit(S) = viol(S) + \frac{obj(S)}{W_{tot} + 1} \tag{2}$$

Here, $viol(S)$ is the number of nodes that have zero neighbors in S (i.e., the number of nodes violating the total domination condition), and W_{tot} is the sum of weights of all nodes and edges in the graph, that is, the trivial upper bound of the problem. Since $\frac{obj(S)}{W_{tot}+1}$ is always less than 1, the fitness value is primarily dependent on the $viol(S)$ term.
Every feasible solution is a total dominating set, so its number of violating nodes must be zero, thus the fitness function value of every feasible solution is strictly lower than 1.
On the other hand, every infeasible solution has at least one violating node, so its fitness function value must be at least 1. Therefore, every feasible solution has fitness lower than the fitness of any infeasible solution. If two solutions have the same number of violating nodes (they are equally infeasible), the one with the lower objective value will be preferred. Consequently, both feasible and infeasible solutions are properly ordered.

3.3. Shaking and moving

The shaking procedure forms a new solution S_{new} by randomly moving the current solution S within the neighborhood of size κ. This movement is essential for bringing diversification in the search process, and its impact is controlled with boundary values for κ, i.e., κ_{min} and κ_{max}.
The proposed shaking uses a node removal strategy, i.e., it removes κ randomly selected nodes belonging to the current solution. Therefore, S_{new} usually has worse fitness value than S. In most cases, the lowered fitness function value is ascribed to an increased violation term. For example, a previously feasible S might become an infeasible S_{new}.
Luckily, local search procedures are further applied in order to move S_{new} to its nearest local optimum, which will most likely improve fitness by repairing an infeasible solution and/or fine-tuning selected nodes within a feasible solution.
After this step, if S_{new} is better than S, we move to a new one. In addition, neighborhood parameter κ is set to its minimal value κ_{min}. If S_{new} has equal fitness as S, S_{new} is accepted with a fair chance (probability 0.5). Otherwise, we keep the current solution and move on to the subsequent (larger) neighborhood. In case κ exceeds the upper boundary $\kappa_{\max}$, it is brought back to κ_{min}.

3.4. Local search

Local search (LS) combines two first-improvement strategies: `LocalSearch1` (or LS1) with time complexity $O(|V|)$ per LS iteration and `LocalSearch2` (or LS2) with time complexity $O(|V|^2)$ per LS iteration. The details of these two local search procedures are given in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2 Function LocalSearch1

Input: S: solution; $G = (V, E, f_1, f_2)$: problem instance.
$improved \leftarrow true$
$currFit \leftarrow fit(S, G)$
while $improved$ **do**
    $improved \leftarrow false$
    for $v \in$ `RandomOrder`(V) **do**
        if $v \in S$ **then**
            $newFit \leftarrow$ `RecalcNodeRemoved`(S, v, G)
            if $newFit < currFit$ **then**
                $S \leftarrow S \setminus \{v\}$
                $currFit \leftarrow newFit$
                $improved \leftarrow true$
                break
            end if
        else // $v \in V \setminus S$
            $newFit \leftarrow$ `RecalcNodeAdded`(S, v, G)
            if $newFit < currFit$ **then**
                $S \leftarrow S \cup \{v\}$
                $currFit \leftarrow newFit$
                $improved \leftarrow true$
                break
            end if
        end if
    end for
end while
Output: S

LS1 is a 1-inversion first improvement strategy, meaning it tries to add a not-included node or to remove an included one. Due to its first improvement approach, there is a positional (ordering) bias when accepting the first improvement. In order to avoid this bias, at the beginning of each LS1 iteration (iteration of the outer loop), nodes ordering is randomly changed (shuffled). Then we iterate through the shuffled node list, and for every node v, we do the following:

- if v is included in the given solution S, we calculate its removal effect;
- oppositely, if v is not included in the solution (i.e. from $V \setminus s$), we calculate the effect of its inclusion.

Whenever the improvement effect is encountered (fitness function value is reduced), iteration is immediately stopped and change is applied.

Since adding or removing a single node has a very local effect on the overall solution structure, we never recalculate fitness function value from scratch. Instead, we use partial (incremental) fitness function (see lines 8 and 16 in Algorithm 2) that is later explained in detail in Section 3.5.

LS2 is a 1-swap first improvement strategy (see Algorithm 3). New candidate solutions are systematically enumerated by removing a single node from the solution and adding another one from outside of the solution, thus effectively swapping their places. We find it beneficial to leave the possibility for just adding (or removing) a node without removing (or adding) another node. In some cases, this is more effective than swapping, although swapping is, in many cases, more powerful (less local in its nature). As in LS1, we also exploit the partial fitness function in LS2 to speed up the search process. Considering its higher computational complexity, LS2 is applied occasionally. We use it only if the following two conditions are fulfilled: (*i*) there has been no fitness function value improvement for at least 100 VNS iterations, and (*ii*) the minimum of ten VNS iterations has passed from the previous LS2 call.

3.5. Partial fitness function

For each node in the graph we store a set of edges between that node and all nodes that are part of the solution. Throughout the entire optimization process, each of these sets is kept ordered with respect to edge weight. This way we can access each external edge of every node in $O(1)$ time. As it was previously stated, the solution structure change that takes place after adding a single node to the solution is local in its nature. When we perform the operation of adding node v to the solution, only node v and its direct neighbors are affected. The fitness function value of the original solution is updated in the following way:

1. The weight of v is added, as it becomes part of the solution.
2. If v has a neighbor in the solution, its external edge weight is subtracted.
3. The internal edge weights of node v are added.
4. For every neighbor v' of node v that does not have a neighbor in the solution, the number of violating nodes is decremented by one (since v' now has a neighbor in the solution) and if v' is not in the solution, its external edge weight is added to the fitness function value (if v' is in the solution, this edge weight is already added in step 3).
5. For every neighbor of v that has a neighbor in the solution, its external edge weight is updated if v becomes its nearest neighbor in the solution.

Analogous reasoning stands for removing a single node from the solution. Due to a high level of technical complexity of these two procedures, their pseudo-codes are available in supplementary material at `https://github.com/StefanKapunac/wtdp_public/`.

4. Experimental evaluation

This section provides empirical evidence of the quality of the proposed VNS. In order to do so, it includes all relevant approaches regarding WTDP available in literature for comparison purposes. In addition, all relevant problem instances are considered and some new ones are introduced. The following four methods for WTDP are compared:

1. The F2 ILP model formulation by Alvarez-Miranda and Sinnl [2], here simply called ILP. Please note that the authors proposed two ILP models (F1 and F2) in the same work; however, we chose to re-implement their F2 model since it performs far better

Algorithm 3 Function LocalSearch2

```
1: Input: S: solution; G = (V, E), f1, f2): problem instance.
2: improved ← true
3: currFit ← fit(S, G)
4: while improved do
5:     improved ← false
6:     for v_out ∈ RandomOrder(V \ S) do
7:         newFit ← RecalcNodeAdded(S, v_out, G)
8:         if newFit < currFit then          // adding without removing already improves
9:             S ← S ∪ {v_out}
10:            currFit ← newFit
11:            improved ← true
12:            break
13:        end if
14:    end for
15:    if improved then
16:        continue
17:    end if
18:    for v_in ∈ RandomOrder(S) do
19:        newFit ← RecalcNodeRemoved(S, v_in, G)
20:        if newFit < currFit then          // removing without adding already improves
21:            S ← S \ {v_in}
22:            currFit ← newFit
23:            improved ← true
24:            break
25:        end if
26:        S ← S \ {v_in}          // removing
27:        for v_out ∈ RandomOrder(V \ S) do          // check adding effect
28:            newFit ← RecalcNodeAdded(S, v_out, G)
29:            if newFit < currFit then          // removing and adding (swap) improves
30:                S ← S ∪ {v_out}
31:                currFit ← newFit
32:                improved ← true
33:                break
34:            end if
35:        end for
36:        if improved then
37:            break
38:        else
39:            S ← S ∪ {v_in}          // putting back previously removed v_in
40:        end if
41:    end for
42: end while
```

than the model F1 on their set of instances (w.r.t. the number of instances solved to optimality).

Note that in the same work, the authors combined F1 and F2 models with a B&C framework, originally called F1$^+$ and F2$^+$, where models F1 and F2 serve as their basis, respectively. F1$^+$ and F2$^+$ obtained better results than corresponding ILPs (F1 and F2) on small-to-middle sized instances. However, we did not have sufficient information to re-implement these specific algorithms nor could we obtain the executable versions of their algorithms. In order to make the comparison as fair as possible, the best results from literature among all these models are included in our comparison whenever applicable. Note that the B&C framework belongs to a class of exact approaches, which makes it hardly applicable to the size of instances we introduce. We therefore believe that the performances of the F2 model on large-sized instances should not be significantly different from the performance of the original F2$^+$ framework.

2. A greedy randomized adaptive search procedure, labeled by GRASP, proposed in [2].
3. A hybrid of GRASP and a Genetic algorithm, labelled by GRASP+GA, also proposed in [2].
4. Our VNS algorithm, described in Section 3.

All algorithms are written in PYTHON and translated/executed by using *PyPy* engine, which is a faster alternative for CPython. The experiments were conducted in a single-core mode on a computer with Intel Core i9-9900KF CPU @3.6GHz with a memory limit of 6GB RAM per execution, under Microsoft Windows 10 Pro OS. Note that GRASP and GRASP+GA were carefully re-implemented since source code or binaries could not be obtained. ILP models were solved using CPLEX 20.1 solver. The maximal time allowed for solving each problem instance was set to 30 minutes, i.e., 1800 seconds except for the social network instances used in Section 5, which used higher time limit.

4.1. Problem instances

We used the sets of problem instances known from the literature, generated randomly in [2]. There are two sets. The first one, labelled `MA` from [36], consists of 45 entries. The number of nodes in `MA` ranges from $|V| = 20$ to $|V| = 100$.
The second benchmark set, called `AMS`, is introduced in [2] in order to analyze the influence of different weight structures on their algorithms. The Erdos-Renyi $G(n, p)$ (or $G(|V|, p)$) model was used to generate the total number of 135 instances. The largest instances from this set were those with $|V| = 125$.
Our work is primarily focused on the comparison of heuristic techniques. Therefore, apart from these known small-to-middle-sized instances, we have generated a set of large instances (denoted as `NEW`) using the same instance generation methodology as our predecessors. The weights of nodes and edges are assigned at random, in the same way as in [2], that is, node weights and edge weights are sampled from $\mathcal{U}\{a, b\}$ and $\mathcal{U}\{c, d\}$, respectively, where $(a, b, c, d) \in \{(1, 50, 1, 10), (1, 25, 1, 25), (1, 10, 1, 50)\}$. The size of the node set $|V|$ is chosen from $\{250, 500, 1000\}$, while density parameter (edge probability) $p \in \{0.2, 0.5, 0.8\}$. Since five graphs are sampled for each of the 27 configurations $\{(1, 50, 1, 10), (1, 25, 1, 25), (1, 10, 1, 50)\} \times \{250, 500, 1000\} \times \{0.2, 0.5, 0.8\}$, the total number of randomly generated graphs in `NEW` set is 135.

The public repository with (re-)implementations of all four methods, all problem instances, including the newly introduced ones, raw execution results, and supplementary material can be found at `https://github.com/StefanKapunac/wtdp_public`.

4.2. Parameter tuning

We have conducted several preliminary rounds of experiments to set up a proper value of control parameters in the VNS algorithm. The following values are used in our experimental evaluation:

- $\kappa_{\min} = 1$;
- $\kappa_{\max} = \min\{20, \frac{|V|}{5}\}$;
- $it_{\max} = 3900$.

Parameter $it_{\max}$ was set to 3900 for the sake of fair comparison with the best heuristic approach from the literature, which is the GRASP+GA approach. The best configuration of GRASP+GA according to [2] used equivalent computational effort during its search process. More precisely, the initial population of 100 individuals was generated by using GRASP. Then, in each of the 20 iterations of GA, all pairs of individuals from the population were used to generate new population members. The number of individuals in the population was set to 20, so the number of possible pairs of individuals was $\frac{20 \cdot 19}{2} = 190$. Thus the total number of VNS iteration equivalents in the GRASP+GA algorithm was $100 + 20 \cdot 190 = 3900$.

Setting $\kappa_{\min}$ to 1 was a reasonable choice for the considered problem instances and execution time frame. Larger minimal neighborhoods involve higher disturbances in the shaking procedure, which can lead to instability and slow convergence. On the other hand, larger $\kappa_{\min}$ might be beneficial when dealing with very large-scale graphs, which is the case when dealing with social network instances solved in Section 5.

The following set of twelve configurations was used when tuning the $\kappa_{\max}$ parameter: $\{\min\{x, y\} | (x, y) \in \{10, 20, 40\} \times \{logn, \sqrt{n}, \frac{n}{5}, \frac{n}{2}\}\}$. We performed tuning on a subset of problem instances composed of a single representative (out of five) for each instance group. Since there was a total of 315 instances, $\frac{315}{5} = 63$ were used for tuning. Concerning the average solution quality across all instances involved in the tuning, the best results were obtained for the configuration with $x = 20$ and $y = \frac{n}{5}$. Therefore, we decided to use these values in the final VNS evaluation.

4.3. Description of the results

Before going into thorough results analysis, we summarize our findings in Tables 1 and 2. Table 1 reports the results of ILP and our VNS approach. The first column shows the instance group; for example, `MA`-20 means the group of instances with $|V| = 20$ of the benchmark set `MA`. The second column provides the number of instances belonging to the respective group, and the third column ($\overline{best}$) reports the average best solution quality, calculated considering all available methods from the literature, including our newly proposed VNS results. The next two blocks report the results of ILP and VNS, respectively. The first block consists of two columns: column $\overline{obj}$ shows the average solution quality for all instances of the respective group, and column $\#opt$ shows the number of instances that are solved optimally. The second block, which relates to VNS, has four columns: column $\bar{t}$ reports average runtime in seconds, column $\overline{obj}$ reports average solution quality, $\overline{pg\%}$ shows an average relative gap of the provided solution quality w.r.t. the best solution (in %), i.e., $(\frac{obj - best}{best}) \times 100$, and column $\#b$ reports the number of instances where the obtained solution matches the best solution.

The following conclusions may be drawn by looking at Table 1:

- ILP approach is able to solve all 45 instances optimally.
- VNS approach is able to deliver the solutions that match the optimal solutions for all 45 instances. The efficiency of our approach is reflected through low average runtimes (the largest one being $19.3s$).

Table 1: VNS optimality validation.

			ILP		VNS			
instances	#	$\overline{best}$	$\overline{obj}$	#*opt*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	#*b*
MA-20	15	45.5	45.5	15	1.6	45.5	0	15
MA-50	15	88.7	88.7	15	5.2	88.7	0	15
MA-100	15	151.8	151.8	15	19.3	151.8	0	15
All	45	95.3	95.3	45	8.7	95.3	0	45

Table 2 summarizes the results of ILP, our VNS, GRASP, and the hybrid GRASP+GA. The first three columns of the table are the same as in Table 1. The next four blocks report the results of ILP, VNS,GRASP, and the hybrid GRASP+GA, respectively. The columns in the ILP block have the same meaning as in Table 1. Similarly, the meaning of columns inside three respective blocks for VNS, GRASP and GRASP+GA is the same as in the block reserved for VNS in Table 1.

The following conclusions may be drawn after considering the results in Table 2:

- ILP can optimally solve all 45 instances in the AMS-75 group and VNS does the same. This is not the case with the other two methods (GRASP+GA solves optimally 39 cases).
- For AMS-100 and AMS-125, ILP optimally solves a high number of instances (70 out of 90). However, concerning the quality of primal solutions, in 87 (out of 90) instances the solutions of VNS are the best (or tied with the best solutions). The second best approach is ILP, followed by GRASP+GA which is able to provide the best overall solution in 66 cases. The weakest is, as expected, the GRASP approach. Similar conclusions may be drawn considering the average solution quality.
- For the instances of the benchmark set NEW (with $|V| \in \{250, 500, 1000\}$), ILP proved to be inapplicable, since it could not solve any of the 135 instances. The VNS approach provides better results than the other two approaches. This is especially noticeable for the largest group of (15) instances with $|V| = 1000$, where the average solution quality obtained by VNS is 3076.8, whereas the second best approach was GRASP+GA with 3193.5. For 107 instances, VNS obtains the best (or matches the best) solution, the second best being GRASP+GA with 64 best solutions. The average runtime of the best two approaches are comparable, as expected.

Table 2: Vns comparison to Grasp and Grasp+Ga.

			Ilp		Vns				Grasp				Grasp+Ga			
instances	#	$\overline{best}$	$\overline{obj}$	#opt	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	#b	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	#b	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	#b
AMS-75	45	421.7	421.7	45	11.5	421.7	0	45	1.3	443.8	4	17	9.9	423.9	0.34	39
AMS-100	45	524.5	524.5	44	20.2	524.6	0.01	44	2.5	539.2	2.3	16	23.1	526.5	0.32	32
AMS-125	45	610.1	610.1	26	31.1	610.4	0.03	43	5	636.9	3.49	12	47.8	611.8	0.23	34
NEW-250	45	1030.9	1149.9	0	198.5	1032.4	0.1	40	25.8	1115.3	7.41	1	196.6	1035.1	0.35	29
NEW-500	45	1763.5	2408.4	0	1068	1770.5	0.29	33	240.5	1908.3	7.42	0	1136.8	1773.9	0.54	24
NEW-1000	45	3059.7	5932.5	0	1813.3	3076.8	0.4	34	1482.2	3319.3	7.75	0	1834.2	3193.5	4.2	11
All	270	1235.1	1841.2	115	523.8	1239.4	0.14	239	292.9	1327.1	5.4	46	541.4	1260.8	1	169

4.4. Detailed results

In this section we discuss the results on per instance basis. The summarized results showed that instances with up to 100 nodes are solved almost optimally by Vns, namely 134 out of 135 instances. Therefore, we will avoid commenting on these instances and focus on the remaining middle-to-large-sized instances consisting of the group of 45 AMS instances with $|V| = 125$, labeled by AMS-125, and all instances belonging to the benchmark set NEW.

Table 3 gives detailed Vns comparison to Ilp, Grasp, and Grasp+Ga approaches for AMS-125 instances. Note that each row now shows the results of a single instance. The first column stands for the name of the instance, in the form AMS-$|V|$-p-c-w-id, where $(|V|, p, c, w)$ provides instance characteristics; please see paper [2] for more information about instance generation. The next column displays the best result for the respective instance, followed by four blocks. The first block reports the results of the Ilp approach with column *obj* reporting solution quality and column *ind.* indicating the algorithm's finishing status. TL means that the time limit was reached, so optimality is not verified, while *opt* denotes that a solution is optimal. The next three blocks report the results of Vns, Grasp and Grasp+Ga. Each of these four blocks consists of four columns reporting the following parameters: runtime in seconds (t), solution quality (obj), percentage gap for the obtained solution w.r.t. the best reported solution on a respective instance (pg), and indicator column (*ind.*). The indicator has three possibilities here: *best* means that the best–known solution for respective instance is found, *opt* means it is not only the best, but also optimal, while empty space means none of the previous two possibilities.

The following conclusions may be drawn from the detailed results displayed in Table 3:

- In all cases, except for the two where $p = 0.2$ (relatively sparse graphs), Vns was able to deliver the best solution (best generalizes optimal). Note that Grasp+Ga does not reach the best solution in 11 cases.
- The largest percentage gap w.r.t. the best-delivered solution in case of Vns was 1.33.
- The advantage of Vns over Grasp+Ga can be seen in the fact that Vns achieved the optimal solution in 25 out of 26 cases, where Ilp could solve the instances successfully. On the other hand, Grasp+Ga did this in 19 cases.

Table 3: Detailed Vns comparison to Grasp and Grasp+Ga for AMS-125 instances.

		Ilp		Vns				Grasp				Grasp+Ga			
instance	*best*	*obj*	*ind.*	*t*	*obj*	*pg%*	*ind.*	*t*	*obj*	*pg%*	*ind.*	*t*	*obj*	*pg%*	*ind.*
AMS-125-0.2-10-50-1	1026	1026	TL	28.8	1026	0	best	2	1112	8.38		24	1026	0	best
AMS-125-0.2-10-50-2	1038	1038	TL	28.1	1038	0	best	2	1069	2.99		22	1038	0	best
AMS-125-0.2-10-50-3	935	935	opt	24.5	935	0	opt	2	1124	20.21		23	947	1.28	
AMS-125-0.2-10-50-4	1050	1050	TL	26.1	1064	1.33		2	1121	6.76		21	1051	0.1	
AMS-125-0.2-10-50-5	974	974	TL	25.7	974	0	best	2	1112	14.17		25	975	0.1	
AMS-125-0.2-25-25-1	720	720	opt	25.7	720	0	opt	2	803	11.53		26	720	0	opt
AMS-125-0.2-25-25-2	746	746	opt	22.8	746	0	opt	2	768	2.95		24	748	0.27	
AMS-125-0.2-25-25-3	715	715	opt	26.9	715	0	opt	2	752	5.17		21	717	0.28	
AMS-125-0.2-25-25-4	701	701	opt	25.4	701	0	opt	2	726	3.57		22	705	0.57	
AMS-125-0.2-25-25-5	684	684	opt	23.4	685	0.15		2	747	9.21		23	697	1.9	
AMS-125-0.2-50-10-1	455	455	opt	23.2	455	0	opt	2	457	0.44		21	455	0	opt
AMS-125-0.2-50-10-2	477	477	opt	22.4	477	0	opt	2	493	3.35		23	477	0	opt
AMS-125-0.2-50-10-3	490	490	opt	22	490	0	opt	2	501	2.24		21	490	0	opt
AMS-125-0.2-50-10-4	467	467	opt	22.5	467	0	opt	2	504	7.92		23	467	0	opt
AMS-125-0.2-50-10-5	457	457	opt	23.1	457	0	opt	2	468	2.41		24	459	0.44	
AMS-125-0.5-10-50-1	817	817	TL	33.8	817	0	best	4	817	0	best	41	817	0	best
AMS-125-0.5-10-50-2	815	815	TL	34.4	815	0	best	5	827	1.47		45	815	0	best
AMS-125-0.5-10-50-3	836	836	TL	35.8	836	0	best	4	880	5.26		45	872	4.31	
AMS-125-0.5-10-50-4	867	867	TL	33.7	867	0	best	4	914	5.42		55	867	0	best
AMS-125-0.5-10-50-5	867	867	TL	28.6	867	0	best	5	906	4.5		55	867	0	best
AMS-125-0.5-25-25-1	566	566	TL	32	566	0	best	5	566	0	best	48	566	0	best
AMS-125-0.5-25-25-2	533	533	opt	28.6	533	0	opt	5	561	5.25		48	533	0	opt
AMS-125-0.5-25-25-3	538	538	opt	30.4	538	0	opt	5	567	5.39		49	538	0	opt
AMS-125-0.5-25-25-4	552	552	TL	31.1	552	0	best	4	565	2.36		53	552	0	best
AMS-125-0.5-25-25-5	545	545	TL	31.7	545	0	best	5	548	0.55		48	548	0.55	
AMS-125-0.5-50-10-1	334	334	opt	29.3	334	0	opt	4	336	0.6		40	334	0	opt
AMS-125-0.5-50-10-2	330	330	opt	28.5	330	0	opt	4	330	0	opt	38	330	0	opt
AMS-125-0.5-50-10-3	315	315	opt	26.5	315	0	opt	5	315	0	opt	49	315	0	opt
AMS-125-0.5-50-10-4	316	316	opt	29.2	316	0	opt	5	316	0	opt	51	316	0	opt
AMS-125-0.5-50-10-5	311	311	opt	28.9	311	0	opt	4	311	0	opt	40	311	0	opt
AMS-125-0.8-10-50-1	793	793	TL	39.5	793	0	best	9	793	0	best	78	793	0	best
AMS-125-0.8-10-50-2	845	845	TL	40.4	845	0	best	8	854	1.07		72	845	0	best
AMS-125-0.8-10-50-3	787	787	TL	41.8	787	0	best	9	829	5.34		74	787	0	best
AMS-125-0.8-10-50-4	777	777	TL	40.8	777	0	best	9	829	6.69		83	777	0	best
AMS-125-0.8-10-50-5	813	813	TL	40.7	813	0	best	8	827	1.72		77	813	0	best
AMS-125-0.8-25-25-1	508	508	opt	37.4	508	0	opt	9	521	2.56		69	510	0.39	
AMS-125-0.8-25-25-2	498	498	opt	38.3	498	0	opt	9	499	0.2		65	498	0	opt
AMS-125-0.8-25-25-3	513	513	TL	37	513	0	best	9	523	1.95		77	513	0	best
AMS-125-0.8-25-25-4	493	493	opt	38.9	493	0	opt	8	506	2.64		75	493	0	opt
AMS-125-0.8-25-25-5	504	504	TL	38.3	504	0	best	8	519	2.98		76	504	0	best
AMS-125-0.8-50-10-1	307	307	opt	33.6	307	0	opt	8	307	0	opt	64	307	0	opt
AMS-125-0.8-50-10-2	296	296	opt	37.6	296	0	opt	8	296	0	opt	57	296	0	opt
AMS-125-0.8-50-10-3	294	294	opt	33.6	294	0	opt	8	294	0	opt	71	294	0	opt
AMS-125-0.8-50-10-4	270	270	opt	34.8	270	0	opt	9	270	0	opt	86	270	0	opt
AMS-125-0.8-50-10-5	278	278	opt	33.6	278	0	opt	9	278	0	opt	77	278	0	opt

Table 4 shows a detailed comparison among Vns, Ilp, Grasp, and Grasp+Ga approaches on NEW-250 instances. The following conclusions can be made:

- Ilp (F2 in this case) could not optimally solve any of the instances as a result of reaching time limit (TL). Moreover, Ilp does not provide a single best solution.
- Vns could not deliver the best result in just 5 (out of 45) cases, while for Grasp+Ga this situation occurred more frequently, namely in 16 cases. This indicates a better scalability of Vns compared to the Grasp+Ga approach.
- As expected, Grasp is the fastest but the weakest in terms of delivered solution quality among the four competitors.

Table 4: Detailed VNS comparison to GRASP and GRASP+GA for NEW-250 instances.

		ILP		VNS				GRASP				GRASP+GA			
instance	$\overline{best}$	$\overline{obj}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*
NEW-250-0.2-10-50-1	1662	1933	TL	111.2	1703	2.47		10.5	1871	12.58		122.4	1662	0	best
NEW-250-0.2-10-50-2	1728	2029	TL	108.4	1728	0	best	10.4	1943	12.44		125.4	1768	2.31	
NEW-250-0.2-10-50-3	1754	1943	TL	96.6	1769	0.86		10.3	1957	11.57		143.6	1754	0	best
NEW-250-0.2-10-50-4	1635	1799	TL	100.8	1635	0	best	10.2	1796	9.85		121	1655	1.22	
NEW-250-0.2-10-50-5	1737	1905	TL	103.6	1737	0	best	10.6	1924	10.77		135.6	1739	0.12	
NEW-250-0.2-25-25-1	1144	1244	TL	102.7	1144	0	best	10.4	1300	13.64		141.5	1157	1.14	
NEW-250-0.2-25-25-2	1135	1210	TL	101.9	1135	0	best	10	1331	17.27		112.9	1139	0.35	
NEW-250-0.2-25-25-3	1132	1197	TL	109.2	1132	0	best	10.6	1314	16.08		120.1	1132	0	best
NEW-250-0.2-25-25-4	1162	1228	TL	107.7	1162	0	best	10.4	1227	5.59		103.5	1162	0	best
NEW-250-0.2-25-25-5	1147	1285	TL	111.3	1147	0	best	10.3	1306	13.86		124.2	1149	0.17	
NEW-250-0.2-50-10-1	749	763	TL	108.2	749	0	best	10.8	806	7.61		101.8	749	0	best
NEW-250-0.2-50-10-2	708	715	TL	102.4	708	0	best	10.1	754	6.5		110.4	708	0	best
NEW-250-0.2-50-10-3	719	720	TL	104.3	719	0	best	10.2	756	5.15		103.7	719	0	best
NEW-250-0.2-50-10-4	680	680	TL	117.9	680	0	best	10	723	6.32		107.7	680	0	best
NEW-250-0.2-50-10-5	758	765	TL	105.9	758	0	best	10	788	3.96		127.9	764	0.79	
NEW-250-0.5-10-50-1	1431	1696	TL	195.2	1431	0	best	24.1	1536	7.34		175.7	1431	0	best
NEW-250-0.5-10-50-2	1468	1834	TL	223.5	1468	0	best	24.1	1622	10.49		168.5	1471	0.2	
NEW-250-0.5-10-50-3	1417	1502	TL	234.8	1417	0	best	24	1579	11.43		205.7	1417	0	best
NEW-250-0.5-10-50-4	1492	1716	TL	208.8	1492	0	best	24.7	1632	9.38		183.2	1518	1.74	
NEW-250-0.5-10-50-5	1474	1798	TL	205	1483	0.61		24.3	1549	5.09		175.6	1474	0	best
NEW-250-0.5-25-25-1	900	1018	TL	204.4	900	0	best	24.2	981	9		182	914	1.56	
NEW-250-0.5-25-25-2	921	1044	TL	195.4	921	0	best	24.1	1005	9.12		171.9	939	1.95	
NEW-250-0.5-25-25-3	896	1100	TL	195	896	0	best	25.6	977	9.04		164.4	896	0	best
NEW-250-0.5-25-25-4	956	1082	TL	237.6	956	0	best	26	1017	6.38		189.5	957	0.1	
NEW-250-0.5-25-25-5	914	1060	TL	219.4	915	0.11		25.7	996	8.97		210.2	914	0	best
NEW-250-0.5-50-10-1	533	535	TL	188.1	533	0	best	25.2	561	5.25		161.8	533	0	best
NEW-250-0.5-50-10-2	554	588	TL	203.2	554	0	best	25	595	7.4		189	554	0	best
NEW-250-0.5-50-10-3	556	557	TL	173.7	556	0	best	24.9	571	2.7		166.8	556	0	best
NEW-250-0.5-50-10-4	558	569	TL	181.7	558	0	best	25.2	558	0	best	206.3	558	0	best
NEW-250-0.5-50-10-5	532	532	TL	186.1	532	0	best	24.6	562	5.64		143.6	532	0	best
NEW-250-0.8-10-50-1	1410	1627	TL	304.7	1410	0	best	42.4	1509	7.02		275.6	1435	1.77	
NEW-250-0.8-10-50-2	1401	1547	TL	275.3	1401	0	best	44.1	1467	4.71		272.1	1401	0	best
NEW-250-0.8-10-50-3	1444	1748	TL	313.8	1444	0	best	43.5	1518	5.12		313.1	1444	0	best
NEW-250-0.8-10-50-4	1420	1738	TL	308.7	1420	0	best	43.5	1441	1.48		294	1420	0	best
NEW-250-0.8-10-50-5	1430	1668	TL	290	1430	0	best	42.5	1520	6.29		269.1	1430	0	best
NEW-250-0.8-25-25-1	919	1006	TL	295.2	919	0	best	41.6	983	6.96		295.7	928	0.98	
NEW-250-0.8-25-25-2	835	931	TL	293.3	835	0	best	42.5	887	6.23		286.4	835	0	best
NEW-250-0.8-25-25-3	914	1072	TL	271.5	919	0.55		41.4	966	5.69		295.3	914	0	best
NEW-250-0.8-25-25-4	846	903	TL	289.6	846	0	best	41.3	893	5.56		268.1	846	0	best
NEW-250-0.8-25-25-5	853	911	TL	285.6	853	0	best	40.9	926	8.56		280.1	853	0	best
NEW-250-0.8-50-10-1	489	490	TL	274.8	489	0	best	41	496	1.43		295.3	489	0	best
NEW-250-0.8-50-10-2	507	522	TL	271.5	507	0	best	41.8	518	2.17		254.1	507	0	best
NEW-250-0.8-50-10-3	498	507	TL	274.8	498	0	best	41.6	513	3.01		323.9	499	0.2	
NEW-250-0.8-50-10-4	489	519	TL	289	489	0	best	43.2	496	1.43		287.1	494	1.02	
NEW-250-0.8-50-10-5	482	508	TL	253	482	0	best	42.6	518	7.47		339.3	482	0	best

Table 5 focuses on NEW-500 instances. The following conclusions can be made:

- ILP runs out of time in all cases. It does not reach any of the best solutions.
- VNS could not deliver best solutions for 12 (out of 45) instances compared to 21 instances where GRASP+GA produced sub-best quality.
- VNS and GRASP+GA are comparable on the graphs which are not very dense ($p = 0.2$). However, when graphs become dense, VNS clearly starts to dominate over GRAPS+GA in terms of solution quality. This could be a consequence of node-removal-based shaking, which better fits denser graphs: a sub-linear number of nodes w.r.t. $|V|$ is expected to constitute the optimal (or at least high-quality) solution.

Table 5: Detailed VNS comparison to GRASP and GRASP+GA for NEW-500 instances.

		ILP		VNS				GRASP				GRASP+GA			
instance	$\overline{best}$	$\overline{obj}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*
NEW-500-0.2-10-50-1	2893	3837	TL	650.8	2968	2.59		89.6	3304	14.21		728	2893	0	best
NEW-500-0.2-10-50-2	2937	3736	TL	767	2947	0.34		88.8	3254	10.79		806.5	2937	0	best
NEW-500-0.2-10-50-3	3082	3839	TL	737.2	3082	0	best	87.9	3425	11.13		940.2	3090	0.26	
NEW-500-0.2-10-50-4	2964	3705	TL	584.9	3063	3.34		88.9	3360	13.36		904.5	2964	0	best
NEW-500-0.2-10-50-5	2928	3965	TL	574.7	2928	0	best	88.7	3197	9.19		1069.2	2942	0.48	
NEW-500-0.2-25-25-1	1884	2383	TL	835.2	1884	0	best	87.1	2130	13.06		667.7	1918	1.8	
NEW-500-0.2-25-25-2	1931	2474	TL	603.1	1941	0.52		86.8	2125	10.05		803.8	1931	0	best
NEW-500-0.2-25-25-3	1898	2225	TL	664.5	1898	0	best	86.1	2064	8.75		913.9	1914	0.84	
NEW-500-0.2-25-25-4	1861	2368	TL	603.7	1937	4.08		87.1	2081	11.82		774.7	1861	0	best
NEW-500-0.2-25-25-5	1861	2285	TL	875.1	1861	0	best	86.8	2091	12.36		926.2	1904	2.31	
NEW-500-0.2-50-10-1	1128	1256	TL	667.7	1128	0	best	87.6	1209	7.18		853.2	1133	0.44	
NEW-500-0.2-50-10-2	1143	1263	TL	719.9	1143	0	best	88.5	1217	6.47		831.2	1159	1.4	
NEW-500-0.2-50-10-3	1108	1288	TL	633.6	1108	0	best	89.4	1182	6.68		627.9	1108	0	best
NEW-500-0.2-50-10-4	1145	1246	TL	715.8	1145	0	best	86.4	1222	6.72		708.2	1149	0.35	
NEW-500-0.2-50-10-5	1186	1301	TL	747.8	1188	0.17		87.1	1221	2.95		821.3	1186	0	best
NEW-500-0.5-10-50-1	2590	3864	TL	1560.1	2590	0	best	231.1	2788	7.64		1031.5	2614	0.93	
NEW-500-0.5-10-50-2	2542	4551	TL	1624.3	2542	0	best	230.2	2825	11.13		1089.1	2631	3.5	
NEW-500-0.5-10-50-3	2547	4759	TL	1432.8	2547	0	best	229.8	2769	8.72		1011.5	2552	0.2	
NEW-500-0.5-10-50-4	2584	3220	TL	1140.7	2596	0.46		230.4	2805	8.55		1112.4	2584	0	best
NEW-500-0.5-10-50-5	2539	3691	TL	821.4	2539	0	best	229.2	2858	12.56		978.2	2539	0	best
NEW-500-0.5-25-25-1	1543	2164	TL	1059.9	1543	0	best	228.8	1670	8.23		1064.9	1566	1.49	
NEW-500-0.5-25-25-2	1574	2137	TL	1069.6	1594	1.27		229.9	1750	11.18		1266.9	1574	0	best
NEW-500-0.5-25-25-3	1547	2121	TL	1093.1	1547	0	best	235.4	1696	9.63		1054.7	1568	1.36	
NEW-500-0.5-25-25-4	1564	2210	TL	986.5	1566	0.13		246.5	1667	6.59		1029.5	1564	0	best
NEW-500-0.5-25-25-5	1567	2343	TL	998.6	1570	0.19		239.9	1726	10.15		1083.5	1567	0	best
NEW-500-0.5-50-10-1	919	1036	TL	785.7	919	0	best	238.2	943	2.61		961.3	919	0	best
NEW-500-0.5-50-10-2	911	1067	TL	761	911	0	best	232.4	950	4.28		874.7	933	2.41	
NEW-500-0.5-50-10-3	906	1020	TL	808.5	906	0	best	235.7	971	7.17		850.5	906	0	best
NEW-500-0.5-50-10-4	904	1091	TL	852	904	0	best	243.9	949	4.98		995.5	904	0	best
NEW-500-0.5-50-10-5	932	1063	TL	986.2	932	0	best	236.5	956	2.58		945.8	938	0.64	
NEW-500-0.8-10-50-1	2500	3385	TL	1466	2500	0	best	403.8	2644	5.76		1479.9	2500	0	best
NEW-500-0.8-10-50-2	2497	3492	TL	1631.9	2499	0.08		408.6	2565	2.72		1524.7	2497	0	best
NEW-500-0.8-10-50-3	2466	3384	TL	1588.3	2466	0	best	401	2612	5.92		1729.6	2513	1.91	
NEW-500-0.8-10-50-4	2511	5055	TL	1552.9	2511	0	best	402.1	2667	6.21		1769.7	2544	1.31	
NEW-500-0.8-10-50-5	2497	4581	TL	1450.3	2497	0	best	407.5	2578	3.24		1790.4	2538	1.64	
NEW-500-0.8-25-25-1	1490	2004	TL	1326.6	1490	0	best	414.4	1616	8.46		1635.6	1490	0	best
NEW-500-0.8-25-25-2	1505	1972	TL	1523.8	1505	0	best	400.9	1605	6.64		1394.3	1509	0.27	
NEW-500-0.8-25-25-3	1470	2048	TL	1332.7	1471	0.07		401.6	1575	7.14		1693.8	1470	0	best
NEW-500-0.8-25-25-4	1489	1986	TL	1369.6	1489	0	best	390.8	1551	4.16		1532.8	1489	0	best
NEW-500-0.8-25-25-5	1496	2055	TL	1534	1496	0	best	391.9	1595	6.62		1469.2	1507	0.74	
NEW-500-0.8-50-10-1	871	1021	TL	1411.7	871	0	best	398.5	916	5.17		1368.9	871	0	best
NEW-500-0.8-50-10-2	853	968	TL	1406.6	853	0	best	398.2	886	3.87		1611.8	853	0	best
NEW-500-0.8-50-10-3	855	1002	TL	1426.6	855	0	best	389.3	873	2.11		1607	855	0	best
NEW-500-0.8-50-10-4	869	985	TL	1254.7	869	0	best	388.8	894	2.88		1510.7	871	0.23	
NEW-500-0.8-50-10-5	872	933	TL	1421.1	872	0	best	390.7	892	2.29		1313.8	872	0	best

Finally, Table 6 presents a detailed comparison for NEW-1000 instances. The conclusions are as follows:

- ILP could provide primal bounds only for graphs which are relatively sparse ($p = 0.2$). In other cases, the model is not even built within the given memory limit of 6GB, this is denoted as ML.
- VNS delivers best solutions for 34 (out of 45) instances, while GRASP+GA does this in only 11 cases.
- It is interesting that for sparser graphs ($p = 0.2$), GRAPS+GA delivers better results than the VNS approach. However, concerning other instances ($p \in \{0.5, 0.8\}$), VNS shows its power by delivering best results in all cases. For these instances, GRASP+GA does not produce a single best solution.

Table 6: Detailed VNS comparison to GRASP and GRASP+GA for NEW-1000 instances.

		ILP		VNS				GRASP				GRASP+GA			
instance	$\overline{best}$	$\overline{obj}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*
NEW-1000-0.2-10-50-1	4990	10851	TL	1802.9	5106	2.32		809.3	5661	13.45		1844.6	4990	0	best
NEW-1000-0.2-10-50-2	5233	9687	TL	1806.7	5316	1.59		824.3	5803	10.89		1912	5233	0	best
NEW-1000-0.2-10-50-3	5044	9995	TL	1800.1	5044	0	best	829	5790	14.79		2030.3	5196	3.01	
NEW-1000-0.2-10-50-4	5079	9760	TL	1803.5	5321	4.76		829.2	5742	13.05		1901.6	5079	0	best
NEW-1000-0.2-10-50-5	5098	10617	TL	1802.4	5260	3.18		811.6	5887	15.48		1817.7	5098	0	best
NEW-1000-0.2-25-25-1	3167	5340	TL	1803.2	3233	2.08		804.9	3525	11.3		1903.5	3167	0	best
NEW-1000-0.2-25-25-2	3163	5845	TL	1804.6	3163	0	best	827.8	3611	14.16		1954.9	3179	0.51	
NEW-1000-0.2-25-25-3	3148	5003	TL	1800.7	3148	0	best	810.7	3520	11.82		1802.9	3159	0.35	
NEW-1000-0.2-25-25-4	3227	4863	TL	1802.4	3268	1.27		808.1	3583	11.03		1839.1	3227	0	best
NEW-1000-0.2-25-25-5	3234	5349	TL	1801	3234	0	best	813.6	3600	11.32		1848.6	3285	1.58	
NEW-1000-0.2-50-10-1	1907	2419	TL	1807.6	1930	1.21		802.7	2072	8.65		1876.6	1907	0	best
NEW-1000-0.2-50-10-2	1882	2275	TL	1803.2	1890	0.43		812.5	2034	8.08		1826.3	1882	0	best
NEW-1000-0.2-50-10-3	1899	2401	TL	1820.8	1921	1.16		802.3	2042	7.53		1851.7	1899	0	best
NEW-1000-0.2-50-10-4	1914	2281	TL	1801.6	1916	0.1		816.2	2133	11.44		1842	1914	0	best
NEW-1000-0.2-50-10-5	1934	2302	TL	1807.3	1936	0.1		807.5	2095	8.32		1883.9	1934	0	best
NEW-1000-0.5-10-50-1	4483	-	ML	1817.9	4483	0	best	1813.7	4918	9.7		1817.8	4869	8.61	
NEW-1000-0.5-10-50-2	4462	-	ML	1802.6	4462	0	best	1805.5	4857	8.85		1814.3	4807	7.73	
NEW-1000-0.5-10-50-3	4567	-	ML	1809.1	4567	0	best	1820.4	4983	9.11		1819.8	5015	9.81	
NEW-1000-0.5-10-50-4	4480	-	ML	1847.1	4480	0	best	1811.4	4855	8.37		1804.8	4967	10.87	
NEW-1000-0.5-10-50-5	4469	-	ML	1812.2	4469	0	best	1822.7	4765	6.62		1805.9	4862	8.79	
NEW-1000-0.5-25-25-1	2743	-	ML	1826.7	2743	0	best	1821.6	2927	6.71		1822.6	2961	7.95	
NEW-1000-0.5-25-25-2	2771	-	ML	1814.8	2771	0	best	1817.7	2914	5.16		1811.8	2905	4.84	
NEW-1000-0.5-25-25-3	2783	-	ML	1813.2	2783	0	best	1820.9	2940	5.64		1815.7	2923	5.03	
NEW-1000-0.5-25-25-4	2757	-	ML	1800	2757	0	best	1812.2	2987	8.34		1812	2942	6.71	
NEW-1000-0.5-25-25-5	2752	-	ML	1800.1	2752	0	best	1822	2949	7.16		1818.3	2946	7.05	
NEW-1000-0.5-50-10-1	1612	-	ML	1801.3	1612	0	best	1801.7	1702	5.58		1810.5	1699	5.4	
NEW-1000-0.5-50-10-2	1631	-	ML	1808	1631	0	best	1816.3	1698	4.11		1814	1697	4.05	
NEW-1000-0.5-50-10-3	1625	-	ML	1801.3	1625	0	best	1804.3	1699	4.55		1819.2	1711	5.29	
NEW-1000-0.5-50-10-4	1629	-	ML	1836	1629	0	best	1821.5	1714	5.22		1805	1710	4.97	
NEW-1000-0.5-50-10-5	1597	-	ML	1825.2	1597	0	best	1814.1	1677	5.01		1814	1694	6.07	
NEW-1000-0.8-10-50-1	4334	-	ML	1806.3	4334	0	best	1803.1	4518	4.25		1827.1	4678	7.94	
NEW-1000-0.8-10-50-2	4352	-	ML	1800.3	4352	0	best	1813.7	4577	5.17		1807.2	4630	6.39	
NEW-1000-0.8-10-50-3	4358	-	ML	1841.2	4358	0	best	1821.3	4623	6.08		1800.8	4698	7.8	
NEW-1000-0.8-10-50-4	4375	-	ML	1818.5	4375	0	best	1829.3	4713	7.73		1817.2	4660	6.51	
NEW-1000-0.8-10-50-5	4373	-	ML	1857.4	4373	0	best	1816.4	4606	5.33		1814	4587	4.89	
NEW-1000-0.8-25-25-1	2580	-	ML	1885	2580	0	best	1828.7	2719	5.39		1802.1	2723	5.54	
NEW-1000-0.8-25-25-2	2605	-	ML	1848.6	2605	0	best	1807.4	2756	5.8		1811.4	2736	5.03	
NEW-1000-0.8-25-25-3	2577	-	ML	1818.6	2577	0	best	1817.6	2752	6.79		1824.2	2742	6.4	
NEW-1000-0.8-25-25-4	2574	-	ML	1815.1	2574	0	best	1818.7	2647	2.84		1803.4	2668	3.65	
NEW-1000-0.8-25-25-5	2613	-	ML	1800.5	2613	0	best	1830.2	2773	6.12		1840.6	2726	4.32	
NEW-1000-0.8-50-10-1	1525	-	ML	1803.7	1525	0	best	1801.2	1594	4.52		1820.5	1597	4.72	
NEW-1000-0.8-50-10-2	1549	-	ML	1800.9	1549	0	best	1803.2	1632	5.36		1819.9	1637	5.68	
NEW-1000-0.8-50-10-3	1526	-	ML	1815.5	1526	0	best	1826.1	1624	6.42		1800.5	1594	4.46	
NEW-1000-0.8-50-10-4	1526	-	ML	1801	1526	0	best	1821	1566	2.62		1801.1	1579	3.47	
NEW-1000-0.8-50-10-5	1541	-	ML	1801.8	1541	0	best	1825.7	1584	2.79		1806.2	1596	3.57	

4.5. Statistical comparisons

For checking the statistical significance of the observed differences of the four presented approaches, we performed the following statistical methodology. First, Friedman's test was separately executed for all four competitor approaches on the AMS and NEW benchmark sets. After that, in cases when the null hypothesis was rejected (H_0 states that competitor approaches are statistically equal) pairwise comparisons were further performed by using the Nemenyi post-hoc test [42].

The outcome is represented within critical difference (CD) plots. A CD plot positions each approach on the horizontal axis according to its average ranking. Then, the CD is computed for a significance level of 0.05. If the difference is small enough, that is, no statistical difference is detected, a horizontal bar connecting statistically (equal) approaches is drawn.

We divided the results into two groups: the first group includes results for AMS instances, and the second for NEW. Their respective CD plots are shown in Figures 1 and 2, which makes it possible to draw the following conclusions:

- Concerning the results on the benchmark set AMS, as the ILP model solves 115 (out

of 135) instances to optimality, it is expected that it has the best average ranking on these small-to-middle-sized instances. A slightly weaker average ranking is obtained by our VNS; however, the remaining two competitors are far behind. There is no statistically significant difference between the results delivered by ILP and VNS. The difference between the results of VNS and other two heuristic approaches is statistically significant in favor of the VNS approach.

- Concerning the results on benchmark set `NEW`, it can be seen that there is a significant difference among all approaches. More precisely, VNS outperforms GRASP+GA, which further outperforms GRASP. Finally, as expected, ILP performs worst due to its inability to deal with larger problem dimensions.

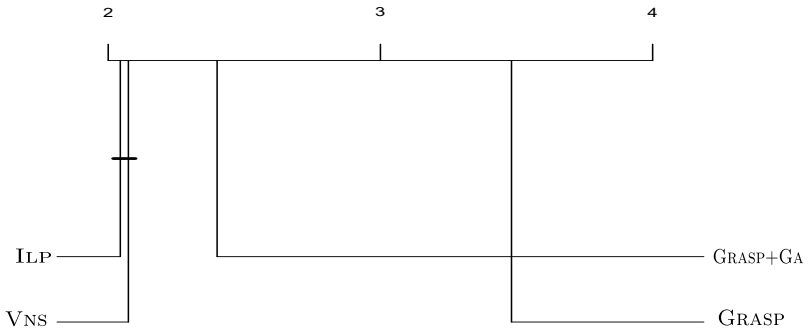


Figure 1: CD plot for the results of all instances from benchmark set `AMS`.

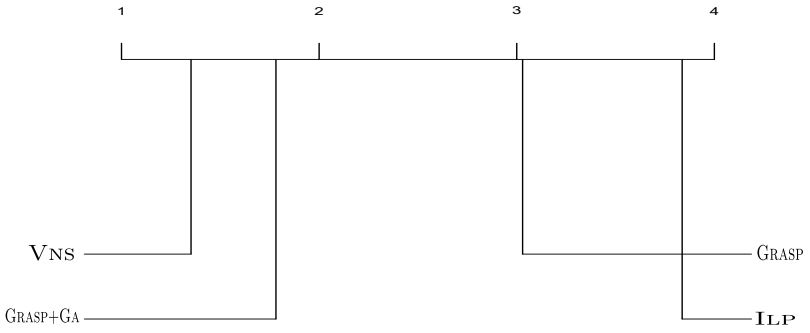


Figure 2: CD plot for the results of all instances from benchmark set `NEW`.

In order to check the statistical differences of the four approaches on all middle-to-large-sized instances, we performed statistical analysis on `AMS` and `NEW` joined together. This statistical evaluation is presented in Figure 3. It can be seen that the best approach overall is VNS. The second best approach is GRASP+GA, followed by ILP, and finally GRASP.

5. Application of WTDP solutions to information spreading in social networks

In this section we demonstrate the impact of initial spreader selection to information spreading in large social networks. Rumors, as a kind of information that can be spread, are a form of communication that can affect the entire society.
The problem of information (rumor, viral marketing information, etc.) spreading is a well-known in data science. Many researchers have attempted to solve it over the last decade.

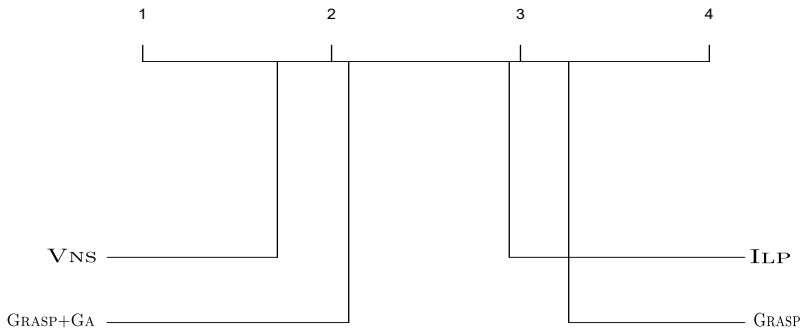


Figure 3: CD plot for the results of all instances from benchmark sets `AMS` and `NEW` combined.

See for example [38, 48, 14], where various SIR-based models have been proposed. SIR model divides the entire population into three groups of individuals: S, I, and R. Although SIR is usually employed to model disease spreading, it also made success in the context of information spreading. For example, in [35], the authors analyze rumor spreading in a microblog network structure; the model for information spreading with the SIR model inside dynamic social networks is considered in [34]; in [29], the authors use SIR to model fake news spreading through WhatsApp.
An example of a commercially appealing application is viral marketing in which users of a social network make person-to-person recommendations, which can effectively lead to an increase in sales and awareness of a commercial product or service. See early research on this matter in [31], which besides other models, also considers SIR. More recent effort is made in [43], where authors conduct comparisons between SIR disease parameters and marketing applications, especially regarding communication marketing campaigns. In [17], the author uses SIR with vector transmission to predict the effectiveness of a viral marketing campaign and the spread of product adoption.

- S generally stands for *susceptible*, or in our case people who have never heard a certain piece of information. These individuals are also called *ignorants* in the context of information spreading.
- I stands for *infective*, in our case people who are spreading the information. They are also called *spreaders*.
- R stands for *recovered*, in our case people who have heard the information but do not spread it. They can be also called *stiflers*.

The rules of information spreading within basic SIR are as follows.

1. When a spreader interacts with an ignorant, the ignorant becomes a spreader with probability β (infectivity rate or spreading rate).
2. When a spreader interacts with another spreader or a stifler, the initiating spreader becomes a stifler with probability γ (recovery rate or immunity rate).

The spreading process starts with a subset of people (nodes) from the network being initially informed and stops when no information spreaders are left in the network.

Let us consider the case where the initial set of spreaders is a WTDP solution, $\beta = 0$ and $\gamma = 1$. All other nodes are ignorants. By definition of a feasible WTDP solution, every

ignorant node in the graph has a spreader neighbor. However, since $\beta = 0$, all ignorants will remain ignorants. Similarly, by definition of a WTDP solution, every spreader node has a neighbor spreader. Since $\gamma = 1$, all spreaders will become stiflers after a single iteration. Since at that point there are no more spreaders in the network, the whole process also ends after only one iteration. When probabilities β and γ are not set to 0 or 1, the process is obviously not deterministic.
Our *main observation* is that setting the initial set of spreaders to a WTDP solution of size k will induce faster information spreading than when the same number of spreaders is randomly selected.

The performed experiments are based on the SNAP collection of social network datasets from [32]. These large-scale instances represent parts of real–world social networks such as Facebook or Twitter, which are of great importance in the domain of information spreading. We used 27 different instances, with sizes ranging from 1912 to 81306 nodes, and 31299 to 1342310 edges.

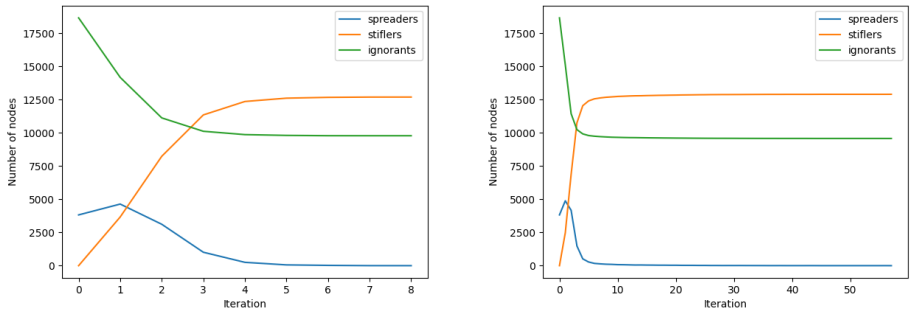
Table 7: Convergence time comparison.

			$(0.1, 0.1)$		$(0.1, 0.5)$		$(0.1, 0.9)$		$(0.5, 0.1)$		$(0.5, 0.5)$		$(0.5, 0.9)$		$(0.9, 0.1)$		$(0.9, 0.5)$		$(0.9, 0.9)$	
instance	$\|V\|$	$\|E\|$	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.
deezer_HR	54573	498202	80	75	10	64	10	49	77	90	11	16	7	10	86	69	14	15	6	7
deezer_HU	47538	222887	71	82	13	70	12	69	63	76	15	13	7	8	87	75	16	16	6	8
deezer_RO	41773	125826	91	93	15	67	10	60	91	71	17	17	8	11	94	80	16	18	6	8
Slashdot0811	77360	546487	48	48	12	10	6	8	48	57	9	11	6	8	46	51	11	10	5	9
Slashdot0902	82168	582533	63	67	16	84	6	50	65	80	12	13	6	9	75	84	15	14	6	7
Wiki-Vote	7115	100762	61	52	12	43	5	30	87	78	11	14	6	8	106	80	14	18	6	6
facebook_artist	50515	819306	58	88	15	43	8	51	72	71	14	20	6	8	75	126	15	16	6	8
facebook_athletes	13866	86858	60	85	12	53	9	29	55	70	12	16	6	8	73	61	14	16	6	7
facebook_company	14113	52310	75	72	14	48	9	48	76	61	11	16	6	15	96	77	13	12	6	8
deezer_europe	28281	92752	71	80	15	53	9	62	104	82	14	16	8	9	81	95	17	15	7	10
facebook_combined	4039	88234	38	28	9	14	11	18	45	96	9	11	6	9	39	39	9	12	4	8
facebook_government	7057	89455	49	47	12	29	11	36	67	41	11	13	6	9	48	61	11	12	6	7
lastfm_asia	7624	27806	67	67	11	54	7	85	94	81	13	19	6	8	68	63	11	14	6	8
musae_DE	9498	153138	37	62	8	41	7	19	66	57	10	14	6	6	52	76	12	13	5	7
musae_ENGB	7126	35324	67	52	10	24	8	40	65	69	11	16	5	7	99	84	12	15	5	8
musae_ES	4648	59382	37	50	9	29	8	41	68	40	10	10	6	7	50	53	10	12	5	9
musae_FR	6549	112666	53	38	8	18	8	7	54	80	13	10	5	6	65	96	12	10	6	7
musae_PTBR	1912	31299	21	43	13	11	6	7	50	61	8	11	5	6	60	56	8	10	5	7
musae_RU	4385	37304	72	42	10	38	7	36	60	59	10	13	5	9	57	65	14	12	5	6
musae_facebook	22470	171002	64	92	12	56	9	47	88	78	11	16	7	9	74	84	12	14	6	9
musae_git	37700	289003	73	71	13	64	7	44	95	90	14	16	6	8	115	85	16	15	7	7
facebook_new_sites	27917	206259	55	60	14	65	9	45	93	53	16	15	8	9	76	106	12	13	6	8
facebook_politician	5908	41729	50	82	12	44	9	47	53	84	12	13	6	11	53	61	11	13	5	9
facebook_public_figure	11565	67114	53	103	15	65	8	60	74	70	15	16	6	12	71	87	15	15	5	9
soc-Epinions1	75879	405740	94	107	14	80	7	139	107	102	19	19	7	15	100	104	18	21	7	10
facebook_tvshow	3892	17262	58	71	10	36	9	45	60	55	12	19	6	12	62	67	12	14	6	7
twitter_combined	81306	1342310	56	53	11	15	13	20	74	116	13	21	7	8	82	80	15	15	6	8
Average	27288	233442.6	60.1	67	12	45.1	8.4	44.1	72.3	72.9	12.3	15	6.3	9.1	73.7	76.5	13.1	14.1	5.7	7.9

Table 7 shows comparison results for each of the 27 instances, along with averaged results in the last row. The comparison metric is the number of iterations needed for the spreading process to finish. We will call this metric *convergence time*. The first three columns show the characteristics of respective instances, namely instance name, number of nodes ($|V|$) and number of edges ($|E|$). Different combinations of probabilities were tested, i.e., $(\beta, \gamma) \in \{0.1, 0.5, 0.9\} \times \{0.1, 0.5, 0.9\}$. For each combination we reserve a single block containing two columns. The first one shows the convergence time when WTDP solution, obtained with VNS, is used for SIR (column V.). The second one shows the convergence time when SIR dynamics is initiated with randomly selected spreading nodes (column R.)
The following observations may be drawn from these numerical results:

- When the spreading rate is low, i.e., $\beta = 0.1$, it is obvious that the WTDP approach provides quicker convergence compared to the case when nodes are selected randomly. In the case ($\beta = 0.1, \gamma = 0.9$) when this is especially noticeable, convergence time is 8.4 versus 44.1. In general, when the spreading rate is low, the choice of initial spreaders is more important, which is natural.
- Contrary, when the spreading rate is very high (which is unrealistic), it becomes less important to have *good* starting spreaders. Even in these cases, initial choices delivered by the VNS WTDP solution are better than randomly selected. However, these differences are not that high anymore, see for example the case when $\beta = 0.9$ and $\gamma = 0.5$, where convergence time is 13.1 versus 14.1.
- Higher immunity rate scenarios also benefit from using the VNS WTDP solution, since this slows down the spreading. This can be empirically proved by looking at relative performances for a fixed β and increasing γ. For example, when $\beta = 0.9$, relative convergence time (average V. divided by average R.) decreases as follows: 0.96, 0.93, 0.72.
- To conclude, low spreading rate combined with high immunity rate is the perfect scenario for benefiting from the VNS WTDP approach.

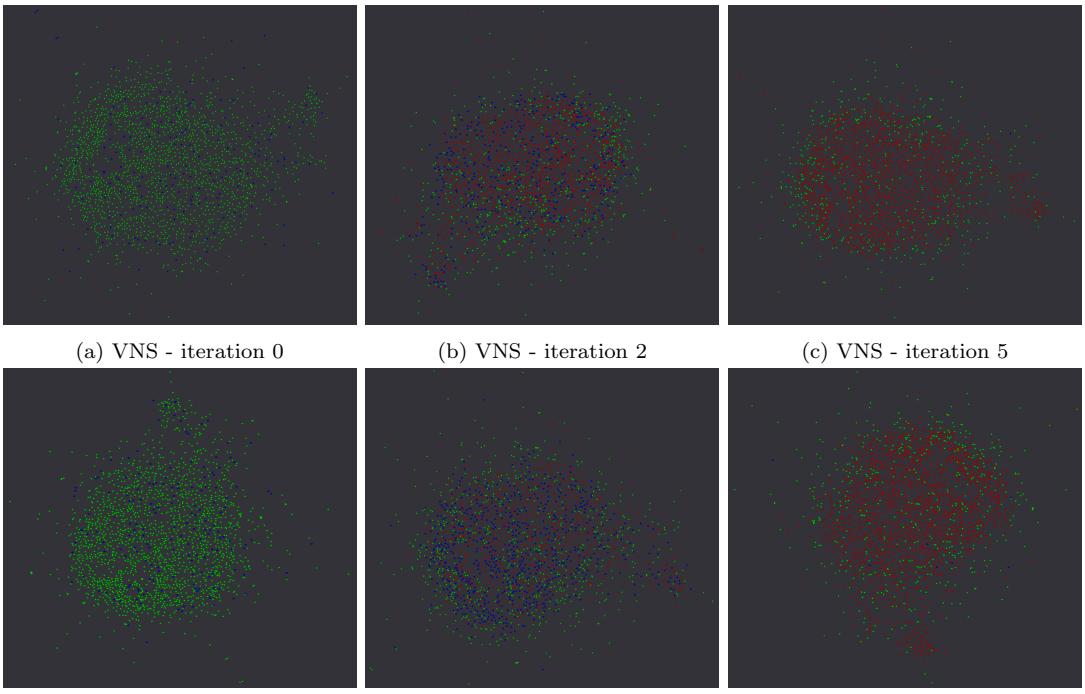
We emphasize that the final distribution of spreaders, stiflers and ignorants is similar upon convergence, as shown in Figure 4. Thus the random selection works quite well when network coverage is in question. The main benefit of the VNS WTDP approach is improved convergence time.



(a) VNS (b) Random

Figure 4: SIR dynamics on musae-facebook instance with $\beta = 0.1$ and $\gamma = 0.9$. Initial set of spreaders is:
(a) a WTDP solution obtained with VNS,
(b) random set of nodes of same cardinality as in (a).
Best viewed in color.

Let us compare the distributions of spreaders, stiflers and ignorants per SIR iteration, demonstrated in Figure 5. At the very beginning, the number of spreaders, stiflers, and ignorants is the same for both approaches. In the second iteration of SIR, there is already a clear advantage of the VNS WTDP approach, i.e., the number of ignorants in the VNS WTDP approach is 913 versus 488 for RANDOM. In the fifth iteration of the VNS WTDP approach there is only one spreader left and the process ends in the following sixth iteration, while the RANDOM approach continues for three more iterations (ending in the eighth iteration).



(a) VNS - iteration 0 (b) VNS - iteration 2 (c) VNS - iteration 5

(d) Random - iteration 0 (e) Random - iteration 2 (f) Random - iteration 5

Figure 5: SIR dynamics through iterations on musae-PTBR instance with $\beta = 0.1$ and $\gamma = 0.9$.
Spreaders are shown in blue, stiflers in red and ignorants in green.
Best viewed in color.

6. Conclusions and future work

This paper explored the weighted total domination problem (WTDP) and proposed variable neighborhood search (VNS) based on (*i*) the shaking mechanism which applies the operation of node removals, (*ii*) two effective first-improvement local search procedures called LS1 and LS2, and (*iii*) a partial calculation of the fitness function. LS1 utilizes 1-inversion neighborhood and runs in linear time, whereas LS2 utilizes 1-swap neighborhood and runs in quadratic time. While the LS1 procedure is executed at each iteration of the algorithm, LS2 executions are performed from time to time, as LS2 is more expensive than LS1. A carefully designed fitness function allowed for the evaluation of both feasible and infeasible solutions, permitting the Vns to search promising solution space regions more thoroughly.
Our computational experiments demonstrated the quality of the proposed Vns. For small-to-middle-sized instances, almost all optimal solutions were reached. Vns outperforms all methods from the literature on middle-to-large-sized instances, which was confirmed by appropriate statistical tests.
Additional experiments were conducted to show the practical relevance of WTDP for information spreading within social networks. The WTDP solutions obtained by Vns served as the set of initially informed nodes (spreaders) in the network, which was followed by SIR model execution. We showed that information spreading convergence time is shorter (better) when initially informed nodes are set to WTDP solutions produced by Vns than when they are set randomly.

Future efforts might take several directions. First, one can solve WTDP by employing various (meta)heuristic approaches, either population-based or single-point-based metaheuris-

tics. It would be interesting to see if some incremental heuristics, such as Beam Search, are applicable to WTDP. Second, complex networks seem to be a fertile ground for applying WTDP-like models and their solutions as one might think of different applications of the WTDP or similar problem formulations. Third, one can come up with theoretical conclusions on WTDP, which may include exact or bounding values for regular graphs, or analysis of expected values for different random graph models such as Erdos-Renyi, small-world networks, or preferential attachment networks. These conclusions can later be integrated into (meta)heuristic approaches to initialize populations, perform more subtle problem-specific local searches, etc.

Acknowledgements

Marko Djukanović's research is partially funded by the Ministry for Scientific and Technological Development, Higher Education and Information Society, Government of Republic of Srpska, B&H under the Project "Development of artificial intelligence methods for solving computer biology problems".

References

[1] R. B. Allan and R. Laskar. On domination and independent domination numbers of a graph. *Discrete mathematics*, 23(2):73–76, 1978.

[2] E. Álvarez-Miranda and M. Sinnl. Exact and heuristic algorithms for the weighted total domination problem. *Computers & Operations Research*, 127:105157, 2021.

[3] A. A. Bertossi. Dominating sets for split and bipartite graphs. *Information processing letters*, 19(1):37–40, 1984.

[4] K. S. Booth and J. H. Johnson. Dominating sets in chordal graphs. *SIAM Journal on Computing*, 11(1):191–199, 1982.

[5] S. Bouamama and C. Blum. An improved greedy heuristic for the minimum positive influence dominating set problem in social networks. *Algorithms*, 14(3):79, 2021.

[6] S. Bouamama, C. Blum, and J.-G. Fages. An algorithm based on ant colony optimization for the minimum connected dominating set problem. *Applied Soft Computing*, 80:672–686, 2019.

[7] S. Cai, W. Hou, Y. Wang, C. Luo, and Q. Lin. Two-goal local search and inference rules for minimum dominating set. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 1467–1473, 2021.

[8] S. Chakravarty and A. Shekhawat. Parallel and serial heuristics for the minimum set cover problem. *The Journal of Supercomputing*, 5(4):331–345, 1992.

[9] M. Chlebík and J. Chlebíková. Approximation hardness of dominating set problems. In *European Symposium on Algorithms*, pages 192–203. Springer, 2004.

[10] E. Cockayne. Domination of undirected graphs—a survey. In *Theory and Applications of Graphs*, pages 141–147. Springer, 1978.

[11] E. J. Cockayne, R. Dawes, and S. T. Hedetniemi. Total domination in graphs. *Networks*, 10(3):211–219, 1980.

[12] P. Corcoran and A. Gagarin. Heuristics for k-domination models of facility location problems in street networks. *Computers & Operations Research*, page 105368, 2021.

[13] M. Damian and S. V. Pemmaraju. Apx-hardness of domination problems in circle graphs. *Information processing letters*, 97(6):231–237, 2006.

[14] G. F. de Arruda, F. A. Rodrigues, and Y. Moreno. Fundamentals of spreading processes in single and multilayer complex networks. *Physics Reports*, 756:1–59, 2018.

[15] R. G. Downey, M. R. Fellows, C. McCartin, and F. Rosamond. Parameterized approximation of dominating set problems. *Information Processing Letters*, 109(1):68–70, 2008.

[16] F. V. Fomin, D. Kratsch, and G. J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 245–256. Springer, 2004.

[17] M. Freed. Using the sir epidemiology model with vector transmission to predict the effectiveness of a viral marketing campaign and the spread of product adoption. *Undergraduate Journal of Mathematical Modeling: One+ Two*, 9(2):1, 2019.

[18] F. Grandoni. A note on the complexity of minimum dominating set. *Journal of Discrete Algorithms*, 4(2):209–214, 2006.

[19] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[20] P. Hansen, N. Mladenović, and J. A. M. Pérez. Variable neighbourhood search: methods and applications. *4OR*, 6(4):319–360, 2008.

[21] P. Hansen, N. Mladenović, and J. A. M. Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[22] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. CRC press, 2013.

[23] A.-R. Hedar and R. Ismail. Hybrid genetic algorithm for minimum dominating set problem. In *International conference on computational science and its applications*, pages 457–467. Springer, 2010.

[24] A.-R. Hedar and R. Ismail. Simulated annealing with stochastic local search for minimum dominating set problem. *International Journal of Machine Learning and Cybernetics*, 3(2):97–109, 2012.

[25] M. A. Henning. A survey of selected recent results on total domination in graphs. *Discrete Mathematics*, 309(1):32–63, 2009.

[26] C. K. Ho, Y. P. Singh, and H. T. Ewe. An enhanced ant colony optimization metaheuristic for the minimum dominating set problem. *Applied Artificial Intelligence*, 20(10):881–903, 2006.

[27] S. Hu, H. Liu, Y. Wang, R. Li, M. Yin, and N. Yang. Towards efficient local search for the minimum total dominating set problem. *Applied Intelligence*, pages 1–15, 2021.

[28] C. Huang, F. Morcos, S. P. Kanaan, S. Wuchty, D. Z. Chen, and J. A. Izaguirre. Predicting protein-protein interactions from protein domains using a set cover approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):78–87, 2007.

[29] P. Khurana and D. Kumar. Sir model for fake news spreading through whatsapp. In *Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT)*, pages 26–27, 2018.

[30] Y. Kikuchi and Y. Shibata. On the domination numbers of generalized de bruijn digraphs and generalized kautz digraphs. *Information Processing Letters*, 86(2):79–85, 2003.

[31] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5–es, 2007.

[32] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[33] C. Liu and Y. Song. Parameterized complexity and inapproximability of dominating set problem in chordal and near chordal graphs. *Journal of combinatorial optimization*, 22(4):684–698, 2011.

[34] C. Liu and Z.-K. Zhang. Information spreading on dynamic social networks. *Communications in Nonlinear Science and Numerical Simulation*, 19(4):896–904, 2014.

[35] J. Liu, K. Niu, Z. He, and J. Lin. Analysis of rumor spreading in communities based on modified sir model in microblog. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 69–79. Springer, 2014.

[36] Y. Ma, Q. Cai, and S. Yao. Integer linear programming models for the weighted total domination problem. *Applied Mathematics and Computation*, 358:146–150, 2019.

[37] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997.

[38] Y. Moreno, M. Nekovee, and A. F. Pacheco. Dynamics of rumor spreading in complex networks. *Physical review E*, 69(6):066130, 2004.

[39] J. Nieminen. Two bounds for the domination number of a graph. *IMA Journal of Applied Mathematics*, 14(2):183–187, 1974.

[40] P. Pinacho-Davidson and C. Blum. Barrakuda: A hybrid evolutionary algorithm for minimum capacitated dominating set problem. *Mathematics*, 8(11), 2020.

[41] P. Pinacho-Davidson, S. Bouamama, and C. Blum. Application of cmsa to the minimum capacitated dominating set problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 321–328, 2019.

[42] T. Pohlert. The pairwise multiple comparison of mean ranks package (pmcmr). *R package*, 27(2019):9, 2014.

[43] H. S. Rodrigues and M. J. Fonseca. Can information be spread as a virus? viral marketing as epidemiological model. *Mathematical methods in the applied sciences*, 39(16):4780–4786, 2016.

[44] J. Rolfes. *Copositive Formulations of the DOMINATING SET Problem and Applications Master Thesis*. PhD thesis, Master's thesis, Department of Mathematics, Faculty of Science, University . . . , 2014.

[45] L. Ruan, D. Du, X. Hu, X. Jia, D. Li, and Z. Sun. Converter placement supporting broadcast in wdm optical networks. *IEEE Transactions on Computers*, 50(7):750–758, 2001.

[46] T.-P. Shuai and X.-D. Hu. Connected set cover problem and its applications. In *International Conference on Algorithmic Applications in Management*, pages 243–254. Springer, 2006.

[47] J. M. Van Rooij and H. L. Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, 2011.

[48] Y. Zan, J. Wu, P. Li, and Q. Yu. Sicr rumor spreading model in complex networks: Counterattack and self-resistance. *Physica A: Statistical Mechanics and its Applications*, 405:159–170, 2014.

[49] X.-Y. Zhang, J. Zhang, Y.-J. Gong, Z.-H. Zhan, W.-N. Chen, and Y. Li. Kuhn–munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks. *IEEE Transactions on Evolutionary Computation*, 20(5):695–710, 2015.