algorithms

MDPI

Article

An Improved Greedy Heuristic for the Minimum Positive Influence Dominating Set Problem in Social Networks

Salim Bouamama [1] and Christian Blum [2,*]

[1] Department of Computer Science, Mechatronics Laboratory (LMETR) - E1764200, Ferhat Abbas University Sétif 1, Sétif 19000, Algeria; salim.bouamama@univ-setif.dz
[2] Artificial Intelligence Research Institute (IIIA-CSIC), Campus of the UAB, 08193 Bellaterra, Spain
* Correspondence: christian.blum@iiia.csic.es

Abstract: This paper presents a performance comparison of greedy heuristics for a recent variant of the dominating set problem known as the minimum positive influence dominating set (MPIDS) problem. This APX-hard combinatorial optimization problem has applications in social networks. Its aim is to identify a small subset of key influential individuals in order to facilitate the spread of positive influence in the whole network. In this paper, we focus on the development of a fast and effective greedy heuristic for the MPIDS problem, because greedy heuristics are an essential component of more sophisticated metaheuristics. Thus, the development of well-working greedy heuristics supports the development of efficient metaheuristics. Extensive experiments conducted on a wide range of social networks and complex networks confirm the overall superiority of our greedy algorithm over its competitors, especially when the problem size becomes large. Moreover, we compare our algorithm with the integer linear programming solver CPLEX. While the performance of CPLEX is very strong for small and medium-sized networks, it reaches its limits when being applied to the largest networks. However, even in the context of small and medium-sized networks, our greedy algorithm is only 2.53% worse than CPLEX.

Keywords: greedy algorithm; minimum positive influence dominating; set problem ; heuristic search; social network

Citation: Bouamama, S.; Blum, C. An Improved Greedy Heuristic for MPIDS Problem in Social Networks. *Algorithms* **2021**, *14*, 79. https://doi.org/10.3390/a14030079

Academic Editor: Frank Werner

Received: 4 February 2021
Accepted: 25 February 2021
Published: 28 February 2021

1. Introduction

Dominating set problems have recently attracted much attention due to their potential application in a variety of real-life settings. Apart from the standard minimum dominating set problem [1,2], examples include the minimum connected dominating set problem [3], the minimum total dominating set problem [4] and the minimum vertex weight dominating set problem [5].

1.1. Problem Background and Motivation

A problem variant that has been studied especially in the context of online social networks is the minimum positive influence dominating set (MPIDS) problem in which a social network is modeled by a simple, connected undirected graph where vertices represent a group of individuals (people) and edges indicate relationships and interactions between them. The problem was first introduced by Wang et *al.* [6], based on the following motivation. With the explosive growth of online social networks, the need for social network analysis tools in order to study the social influences and interactions between individuals within groups and organizations has become a primary concern. As an example, one of the most popular online social networks worldwide is Facebook. In January 2021, Facebook had approximately 2.74 billion users [7], more than any other social network. In addition, ideas and information propagated in social networks can have a significant impact on society (negative or positive) and on various aspects of the life of people. Social norms theory has shown that the behavior of individuals can be affected by perceptions of others' thoughts and behaviors [8]. Thus, exploiting the relationships among people in social networks can provide great benefits to both economy and society. The aim of the MPIDS problem is to identify a small subset of key influential individuals to speed up the spread of positive influence. It can be applied in viral marketing, which is an advertising strategy that utilizes social relationships, such as friendships, professional interactions and families to spread the awareness about a promoted product among individuals in a given social network [9,10]. The idea is to identify a limited number of customers that can quickly lead to the entire network being influenced to adopt the product. Other applications of the MPIDS problem can be found in e-learning software [11], online business [12], drinking, smoking, and drug related problems [6].

1.2. Problem Description and Existing Work

In technical terms, the MPIDS problem can be described as follows. Given a simple, connected undirected graph $G = (V, E)$ it requires to find a dominating set of minimum cardinality such that at least half of the neighbors of each vertex form part of the dominating set. However, the problem was shown to be APX-hard [13]. Note that a problem is said to be APX-hard if there is a polynomial time reduction scheme from every problem in APX to that problem. Moreover, if a problem is APX-hard, it is also NP-hard. Therefore, most of the past and current research efforts concerning the MPIDS problem are focused on greedy heuristics and on some evolutionary approaches. Greedy heuristics [14] are procedures that generate a solution step by step, making a locally optimal choice at each stage based on a so-called greedy function. We briefly review the existing approaches in the following. The first greedy algorithm for the MPIDS problem, referred to as Wang's greedy algorithm [13], is a $H(\Delta)$-approximation algorithm with $O(n^3)$ time complexity, where $n = |V|$, Δ is the maximum vertex degree, and H is the harmonic function. Another greedy algorithm, referred to as Raei's greedy, was published in [15]. This algorithm requires $O(n^2)$ time. It differs from the previous one in the way in which the next vertex at each construction step is chosen. That is, they differ with respect to the used greedy function. An improved version of Wang's greedy, referred to as Fei's greedy, was proposed in [16]. It incorporates a tie-breaking strategy based on Raei's greedy. More recently, Pan et *al.* [17] presented a fast greedy heuristic with a complexity of $O(n \lg n + m)$ that outperforms all previous greedy approaches both in terms of solution quality and computational time. Therefore, Pan's greedy is considered as the currently best-performing greedy algorithm for the MPIDS problem.

To the best of our knowledge, there are only two studies that have attempted to solve the MPIDS problem using metaheuristic approaches. Metaheuristics are approximate techniques for solving hard optimization problems of different types. They are among the most popular algorithms in the context of problem instances that are too large (or too complex) to be solved by exact techniques. Many metaheuristics are built upon subordinate algorithmic components such as greedy heuristics and local search algorithms. Examples of such metaheuristics include simulated annealing, tabu search, iterated local search, and greedy randomized adaptive search procedures. However, the family of metaheuristics also includes a whole range of bio-inspired techniques such as ant colony optimization, genetic and evolutionary algorithms, and particle swarm optimization. Especially in combinatorial optimization, metaheuristics have had considerable success in application areas such as scheduling, routing, bioinformatics, medical research, passenger and freight terminal operations, and data classification. We refer the interested reader to [18,19] for further information. The MPIDS problem was first tackled by a memetic algorithm [20], called ILPMA, which uses tabu search for improving solutions. The second one is a hybrid approach, referred to as HSIA, that combines a genetic algorithm with particle swarm optimization. Both ILPMA and HSIA share two common features: they incorporate a greedy randomized adaptive algorithm similar to GRASP [21] to seed the initial population with good solutions and their performances are compared with those of the greedy algorithms.

1.3. Motivation and Contribution

Wang's greedy, Raei's greedy and Fei's greedy suffer from a common drawback that they are time-consuming and become highly ineffective with an increasing graph size. This is mainly due to the vertex selection strategy employed at each construction step of the solution construction process. In particular, the choice of the next vertex to be included in the current partial solution requires the evaluation of the corresponding greedy function for all vertices that do not form part of the current partial solution, which requires $O(n)$ of time. This time complexity is reduced to $O(\Delta)$ time in the case of both Pan's greedy and our proposal by considering only the neighbors of a particular vertex at each construction step. We expect our algorithm to outperform the currently best greedy algorithm (Pan's algorithm) due to the following reasons. First, we develop a graph pruning procedure that identifies vertices that must form part of an optimal solution. Second, the vertex selection strategy of our algorithm benefits from the exploitation of two greedy functions (*cover-degree* and *need-degree*), which is in contrast to Pan's greedy which only makes use of *cover-degree*. Finally, in a post-processing step we remove redundant vertices.

Our motivation for the development of a fast and effective greedy algorithm is as follows. The development of high-performing metaheuristics for the MPIDS problem is still a challenge. However, the performance of metaheuristics depends largely on the performance of their main components. Greedy heuristics are a key component of many metaheuristics. They are used for generating initial solutions or for reconstructing partial solutions. The contribution of this paper is to review the available literature on greedy heuristics for the MPIDS problem and make a performance comparison among them both in terms of solution quality and computation time. Furthermore, designing an efficient greedy approach is still a relevant challenge despite many prior attempts. Therefore, we also propose an improved greedy algorithm which outperforms the existing greedy approaches both on benchmark instances that have already been considered in the literature, but also on larger complex networks with millions of nodes. Our approach can be considered a further improvement of the basic idea of Pan's work [17].

1.4. Structure of the Paper

The rest of this paper is organized as follows. In Section 2 we give a technical description of the MPIDS problem. In Section 3, we review the available literature on greedy heuristics applied to the MPIDS problem. The proposed greedy algorithm is described in Section 4. In Section 5, we present and discuss the experimental results. Finally, Section 6 summarizes the work and offers directions for future work.

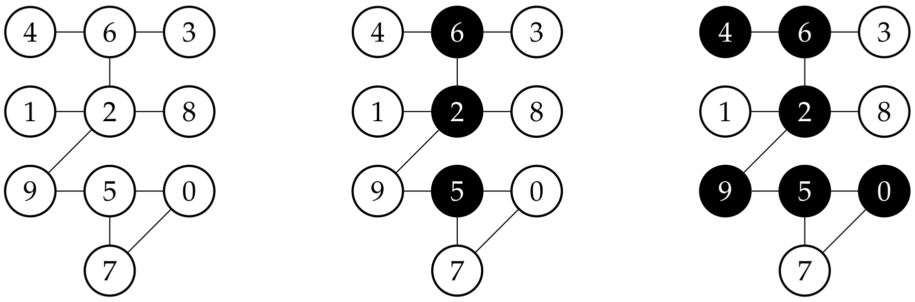
2. The Minimum Positive Influence Dominating Set Problem

We describe the MPIDS problem by starting with basic notions and underlying definitions which are used throughout this paper. Let $G = (V, E)$ be an undirected graph that is both simple and connected. (Note that an undirected graph is called *simple* if it does not contain multiple edges between pairs of vertices and loops.) Hereby, V is a set of n vertices—representing, for example, people—and $E \subset V \times V$ is a set of m edges modeling, for example, relationships between those people. Let v and u be two distinct vertices from G. Now, v and u are said to be adjacent (neighbors) if they are connected by an edge. Further, the *open neighborhood* of v is defined as $N(v) := \{u \in V \mid (v, u) \in E\}$. $N(v)$ is often simply called the neighborhood of v in G. The *closed neighborhood* of v is defined as $N[v] := N(v) \cup \{v\}$,i.e., $N[v]$ includes all vertices adjacent to v, including v itself. The degree of v, denoted by $deg(v)$, is the number of v's neighbors, that is, $deg(v) = |N(v)|$. A vertex of degree one—that is $|N(v)| = 1$—is called a *pendant vertex*.

Definition 1 (Dominating set)**.** *A dominating set (DS) of G is a subset $D \subseteq V$ such that each vertex $v \in V \setminus D$ is adjacent to at least one vertex in D.*

Definition 2 (Positive influence dominating set)**.** *A positive influence dominating set (PIDS) of G is a dominating set $D \subseteq V$ such that each vertex $v \in V$ has at least half of its open neighbors in D, that is, at least $\lceil deg(v)/2 \rceil$ vertices of $N(v)$ must belong to D for each $v \in V$.*

Figure 1b shows an example of a DS of the graph shown in Figure 1a, where black vertices represent those that are in the DS. Figure 1c provides an example of a PIDS (black vertices). Vertex 6, for example, has degree 3 and $\lceil 3/2 \rceil = 2$ of its neighbors (vertex 4 and vertex 2) are part of the PIDS.



(a) A simple, connected undirected graph **(b)** A dominating set **(c)** A positive influence dominating set

Figure 1. An illustrative example of the MPIDS problem. Black vertices form part of the solutions.

Definition 3 (Minimum dominating set problem)**.** *Given an undirected simple graph $G = (V, E)$, the minimum dominating set (MDS) problem asks to find a DS of G of minimum cardinality.*

This problem can be easily formulated in terms of an integer linear program (ILP). For convenience, we assume that V is enumerated as $v_1, v_2, \ldots, v_n$. A binary variable $x_i \in \{0, 1\}$ is associated to each vertex $v_i \in V$ such that $x_i = 1$ if and only if v_i is part of the optimal solution, and $x_i = 0$ otherwise. The ILP model can then be stated as follows.

$$\textbf{Minimize} \quad \sum_{i=1}^{n} x_i \tag{1}$$

$$\textbf{Subject to} \quad \sum_{v_j \in N[v_i]} x_j \geq 1 \quad \forall v_i \in V \tag{2}$$

$$x_i \in \{0, 1\} \tag{3}$$

Equation (2) ensures that the generated solution is a dominating set. Remember that $N[v_i]$ is the closed neighborhood of v_i.

Definition 4 (Minimum positive influence dominating set problem)**.** *Given an undirected simple graph $G = (V, E)$, the minimum positive influence dominating set (MPIDS) problem asks to find a PIDS of G of minimum cardinality.*

Based on the ILP model of the MDS problem from above, the MPIDS problem can also easily be stated in terms of an ILP as follow.

$$\textbf{Minimize} \quad \sum_{i=1}^{n} x_i \tag{4}$$

$$\textbf{Subject to} \quad \sum_{v_j \in N(v_i)} x_j \geq \left\lceil \frac{deg(v_i)}{2} \right\rceil \quad \forall v_i \in V \tag{5}$$

$$x_i \in \{0, 1\} \tag{6}$$

Equation (5) ensures that a feasible solution contains at least half of the neighbors of each vertex $v_i \in V$.

3. Greedy heuristics

Greedy algorithms are very common techniques for constructing solutions to combinatorial optimization problems from scratch, in a step-by-step manner. They are either used as standalone algorithms, or as subordinate algorithmic components of more sophisticated metaheuristics. Algorithm 1 shows the pseudo-code of a basic greedy algorithm for the MPIDS problem. It takes as input a simple, connected undirected graph $G = (V, E)$ representing an instance of the MPIDS problem and provides as output a subset of vertices $S \subset V$ corresponding to a positive influence dominating set. This algorithm builds a solution step by step, starting from an empty solution $S = \emptyset$. At each construction step, it adds one feasible vertex $v^* \in \overline{S}$ to S until a valid solution is obtained. $\overline{S} = V \setminus S$ denotes the set of vertices of V not belonging to S. Note that this set initially contains all the vertices of the graph.

Algorithm 1 MPIDS_Greedy()

```
Input: a simple, connected undirected graph $G = (V, E)$
Output: a positive influence dominating set $S$
1: $S \leftarrow \emptyset$
2: while ($S$ is not a valid PIDS solution) do
3:    $v^* \leftarrow$ argmax{greedy_function($v$) $| v \in \overline{S}$}
4:    $S \leftarrow S \cup \{v^*\}$
5: end while
6: return $S$
```

At each construction step, the choice of the solution component to be added to the current partial solution is made deterministically based on a so-called *greedy function* which plays an important role for the performance of the algorithm. It evaluates each solution component by measuring the local improvement obtained by adding the corresponding component to the incumbent partial solution. Most of the existing greedy algorithms for the MPIDS problem are based on at least one of the two greedy functions that are known as *cover-degree* and *need-degree*. To define them, let $S \subset V$ be the incumbent partial solution which is not yet a PIDS and $h_S(v) = \lceil \frac{deg(v)}{2} \rceil - |N_S(v)|$ where $N_S(v) = N(v) \cap S$ denotes the set of neighbors of $v \in V$ belonging to S. Then, we say that v is covered if $h_S(v) \leq 0$ and not covered otherwise. Consequently, S is a valid solution (i.e., a PIDS) if and only if all vertices of V are covered. Now, for a given $v \in \overline{S}$, *cover-degree* and *need-degree* are calculated as in Equation (7) and Equation (8), respectively.

$$cover\text{-}degree(v) = |\{\, u \in N(v) : h_S(u) > 0\}| \tag{7}$$

$$need\text{-}degree(v) = \sum_{u \in N(v)} \max(h_S(u), 0) \tag{8}$$

The first one represents the number of uncovered neighbors of v with respect to S, whereas the second represents the total need of the uncovered neighbors of v. Table 1 indicates which ones of these two greedy functions were used in the previous works from the literature. In the case in which an algorithm makes use of both functions, one of them was used as a secondary criterion for breaking ties in those cases in which the principal criterion provides the same value for several vertices. In the following, we recall the most recent and best-performing greedy algorithm, that is, Pan's greedy [17]. Note that, even though Pan's greedy uses the same greedy function (*cover-degree*) as Wang's greedy, its time complexity is much lower. This is for the following reason. While Wang's algorithm considers at each construction step any so-far unselected vertex for inclusion into the current partial solution, Pan's algorithm—after choosing the next unselected vertex v in ascending order with respect to the degree—only considers so-far unselected neighbors of v in subsequent construction steps until v is covered.

Table 1. Greedy functions used in the existing literature.

Algorithm Name	Algorithm Type	Greedy Function	Complexity	Year	Ref
Wang's algorithm	Greedy	*cover-degree*	$O(n^3)$	2011	[13]
Raei's algorithm	Greedy	*need-degree*	$O(n^2)$	2012	[15]
Fei's algorithm	Greedy	both	-	2016	[16]
Pan's algorithm	Greedy	*cover-degree*	$O(nlgn + m)$	2019	[17]
ILPMA	A hybrid metaheuristic (GA + TS)	*cover-degree*	-	2018	[20]
HSIA	A hybrid metaheuristic (GA + PSO)	*cover-degree*	-	2019	[22]

The detailed pseudo-code for Pan's algorithm is provided in Algorithm 2. This pseudo-code is the same as in the original publication, but with using our notation and after removing some unnecessary details in order to improve the readability of the algorithm.

Algorithm 2 Pan's greedy algorithm [17]

```
Input: a simple, connected undirected graph $G = (V, E)$
Output: a positive influence dominating set $S$
1: Rename the vertices from $V$ such that $\{v_1, v_2, \cdots, v_n\}$ are the vertices in ascending
   order of the degree
2: $S \leftarrow \emptyset$
3: $\overline{S} \leftarrow V \setminus S$
4: for $i = 1$ to $n$ do
5:   if $h_S(v_i) > 0 : v_i$ is an uncovered vertex then
6:     $\rho \leftarrow h_S(v_i)$
7:     for $j = 1$ to $\rho$ do
8:       $u^* \leftarrow$ argmax$\{cover\text{-}degree(u) \mid u \in N_{\overline{S}}(v_i)\}$
9:       $S \leftarrow S \cup \{u^*\}$
10:      $\overline{S} \leftarrow V \setminus S$
11:    end for
12:  end if
13: end for
14: return $S$
```

4. The Proposed Algorithm

In the following, we describe our improved greedy algorithm—henceforth referred to as IGA-PIDS—which is based on Pan's algorithm for the MPIDS problem.

4.1. The Greedy Procedure

The pseudo-code of IGA-PIDS is given in Algorithm 3. It receives as input a problem instance that consists of a simple, connected undirected graph $G = (V, E)$ with n vertices, and works as follows. First, the given graph is pruned by applying procedure GraphPruning(G) (see Algorithm 4) which returns a partial solution S. This procedure identifies vertices that must form part of the optimal solution and adds them to S. This is done in order to speed up the process of solution construction of the greedy algorithm. Suppose that $v \in V$ is a pendant vertex, i.e., $deg(v) = 1$, and let u be its unique neighbor. First, u is then added to S, because v must be covered and can only be covered by u. Second, if the degree of u is two—that is, $deg(u) = 2$, let $w \neq v$ be the second neighbor of u. As u must covered by exactly one vertex, and as no other vertex benefits from choosing v for covering u, we can savely choose w for covering u. Therefore, w is added to S.

At this point, let C be the set of all so-far uncovered vertices. Next, the algorithm picks yet uncovered vertices from C in (increasing) order of their degrees. The chosen vertex v_i is then covered by choosing neighbors of v_i from $N_{\overline{S}}(v_i)$ until v_i is covered. In particular, at each step the vertex with the largest *cover-degree* value of all vertices in $N_{\overline{S}}(v_i)$ is chosen. If tie breaking is necessary, it is done with the *need-degree* function (see lines 8 and 9 of Algorithm 3). Finally, the algorithm is terminated when all vertices are covered, that is, when S corresponds to a valid PIDS solution.

Algorithm 3 IGA-PIDS: Improved greedy algorithm for the MPIDS problem

```
Input: a simple, connected undirected graph $G = (V, E)$
Output: a positive influence dominating set $S$
 1: $S \leftarrow$ GraphPruning(G)
 2: $C \leftarrow$ set of all so-far uncovered vertices
 3: Rename the vertices from $C$ such that $\{v_1, v_2, \cdots, v_{|C|}\}$ are the vertices in ascending
    order of the degree
 4: $\overline{S} \leftarrow V \setminus S$
 5: while $S$ is not a valid PIDS solution do
 6:   Let $v_i$ be an un-covered vertex with the smallest sub-index in $C$
 7:   $\rho \leftarrow h_S(v_i)$
 8:   for $j = 1$ to $\rho$ do
 9:     $cd_{max} \leftarrow \max\{cover\text{-}degree(u) \mid u \in N_{\overline{S}}(v_i)\}$
10:     $u^* \leftarrow \text{argmax}\{need\text{-}degree(u) \mid u \in N_{\overline{S}}(v_i) \wedge cover\text{-}degree(u) = cd_{max}\}$
11:     $S \leftarrow S \cup \{u^*\}$
12:     $\overline{S} \leftarrow V \setminus S$
13:   end for
14:   $C \leftarrow C \setminus \{v_i\}$
15: end while
16: Reduce(S)
17: return $S$
```

Algorithm 4 Function GraphPruning(G)

```
Input: a simple, connected undirected graph $G = (V, E)$
Output: a partial solution $S$
 1: $S \leftarrow \emptyset$
 2: for each pendant vertex $v \in V$ do
 3:   Let $u$ be the unique neighbor of $v$
 4:   if $u \notin S$ then
 5:     $S \leftarrow S \cup \{u\}$
 6:   end if
 7:   if $deg(u) = 2$ and $v \notin S$ then
 8:     Let $w \neq v$ be the second neighbor of $u$
 9:     if $w \notin S$ then
10:       $S \leftarrow S \cup \{w\}$
11:     end if
12:   end if
13: end for
14: return $S$
```

4.2. Removing Redundant Vertices

The results produced by our greedy algorithm may contain redundant vertices. A redundant vertex is a vertex that can be removed from a solution without making the solution invalid. Formally, a vertex $v \in S$ is redundant if each vertex u from its closed neighborhood $N(v)$ has strictly more than half of its neighbors in S. In other words, $v \in S$ is redundant if and only if $\forall u \in N(v) : h_S(u) < 0$. That is, each vertex in S is checked in sequence to find whether it is redundant. If it is the case, this vertex may safely be removed from S and all values $h_S(.)$ of its neighbors will be decreased by one. A pseudo-code of the complete removal-function is presented in Algorithm 5. This function has a time complexity of $O(n+m)$. Note that our algorithm, by default, makes use of this function. However, we will also test our algorithm without removing redundant vertices. The resulting algorithm variant is henceforth labelled IGA-PIDS^{-}.

Algorithm 5 Function Reduce(S)

```
Input: a valid solution $S$ that may contain redundant vertices
Output: a valid solution $S$ without redundant vertices
1: for each vertex $v \in S$ do
2:   if $h_S(u) < 0$ for all $u \in N(v)$ then
3:     $S \leftarrow S \setminus \{v\}$
4:     for all $u \in N(v)$ do
5:       $h_S(u) \leftarrow h_S(u) - 1$
6:     end for
7:   end if
8: end for
9: return $S$
```

Another way to implement this procedure, which is not considered here, would be to adopt the same removal strategy as presented in [23] for the minimum weight dominating set problem. For the latter, all redundant vertices are initially grouped in a unique set called S^r. Then, a vertex from S^r is iteratively chosen to be removed according to a particular greedy function. In the case of the MPIDS problems, we could select the vertex with the smallest degree, for example. This iterative process stops once S^r becomes empty and all redundant vertices are removed.

4.3. Complexity

Here, we describe the time complexity of IGA-PIDS presented in Algorithm 3. We assume that the problem instance $G=(V,E)$ is represented by an adjacency list and the solution S as a list. It obliviously that function GraphPruning$(\cdot)$ has complexity $O(n)$. Line 3 is done in $O(n \lg n)$ time as $|E| \leq |V|$ and $n=|V|$ is the number of vertices in G. Since the size of S does not exceed n, all other lines, excluding lines 9 and 10, can be done in $O(n)$ time. The running time of lines 9 and 10 is only proportional to the sum of degrees of all vertices in V. The *hand shaking lemma* [24] states that $\sum\limits_{v\in V} deg(v) = 2m$, where $m=|E|$ is the number of edges in G. Therefore, these two lines require $O(m)$ time. In the light of the above, we can conclude that the time complexity of IGA-PIDS takes at most $O(n \lg n + m)$ time.

5. Experimental Evaluation

The proposed greedy algorithm (IGA-PIDS) is compared with four existing greedy algorithms, namely Wang's greedy [13], Raei's greedy [15], Fei's greedy [16] and Pan's greedy [17].

5.1. Computational Setting

In order to conduct a fair comparison with these greedy algorithms, all of them were implemented in ANSI C++ using Cygwin GCC 4.4 for compiling the software and carried out on the same computing platform. The experiments were performed on a laptop equipped with a 64-bit 2.5-GHz Intel® Core™ i5-7200U processor and 8 GB of RAM. Moreover, in order to get an impression about the solution quality provided by IGA-PIDS, we applied the ILP solver ILOG CPLEX 12.10 in single-threaded mode—with a time limit of 2 CPU hours per instance—to all problem instances. Note that the default optimality gaps of CPLEX were used. These default values indicate CPLEX to stop when an integer feasible solution is reached that is within 0.01% of optimality. The experiments with CPLEX were performed on a cluster of machines with two Intel® Xeon® Silver 4210 CPUs with 10 cores of 2.20GHz and 92 Gbytes of RAM.

5.2. Problem Instances

We use three sets of benchmark instances for the experimental evaluation. The first two have already been used by other studies on the MPIDS problem, while the last one (SNAP networks) consists of large-scale complex networks that are studied for the first time in the context of the MPIDS problem.

1. Small social networks: this class of instances contains four well-known real and synthetic networks namely *American College Football* (Football) [25], *Zachary's Karate Club* (Karate) [26], the *Dolphins Network* (Dolphins) [27] and the *Jazz Network* (Jazz) [28]. Characteristics such as the number of vertices and the number of edges of these networks are provided in Table 2. The values of the optimal solutions for each instance (except for the last one) were taken from [20] in which the authors also made use of CPLEX.
2. Large-scale social networks: this class of instances contains 13 real social networks which were provided by the authors of [20] and some of them were originally downloaded from the Network Data Repository [29]. Their characteristics, together with a brief description, are given in Table 3.
3. SNAP social networks: this class of instances contains 22 real complex networks with sizes ranging from 10^4 vertices to $3 \cdot 10^7$. It was download from the Stanford Large Network Dataset Collection [30]. All these instances were originally directed graphs and they are transformed to undirected graphs by neglecting arc orientations and considering parallel edges as one edge. Table 4 gives a brief description of the SNAP networks used in our experiments after preprocessing.

Table 2. Small real social networks used in experiments.

Network	n	m	Opt [20]	Description
Karate	34	78	15	Social network of friendships in a Karate club
Dolphins	62	159	30	Dolphin social network
Football	115	613	63	Network of American college football teams
Jazz	198	2742	-	Collaboration network between Jazz Musicians

Table 3. Large-scale real social networks used in experiments.

Network	n	m	Description
CA-GrQc	5241	14,484	Collaboration network of Arxiv General Relativity
CA-HepTh	9875	25,973	Collaboration network of Arxiv High Energy Physics Theory
CA-HepPh	12,006	118,489	Collaboration network of Arxiv High Energy Physics
CA-AstroPh	18,771	198,050	Collaboration network of Arxiv Astro Physics
CA-CondMat	23,133	93,439	Collaboration network of Arxiv Condensed Matter
Email-Enron	36,692	183,831	Email communication network from Enron
ncsrrlwg2	6396	15,872	Collaboration network between by scientists
actors-data	10,042	145,682	Collaboration network between by actors
ego-facebook	4039	88,234	Social circles from Facebook
socfb-nips-ego	2888	2981	Social friendship network extracted from Facebook
socfb-Mich67	3748	81,903	Social friendship network extracted from Facebook
socfb-Brandeis99	3898	137,567	Social friendship network extracted from Facebook
soc-gplus	23,628	39,194	Social network extracted from Google+

Table 4. SNAP networks used in experiments.

Network	*n*	*m*	**Description**
amazon0302	262,111	899,792	Amazon product co-purchasing network from March 2, 2003
amazon0312	400,727	2,349,869	Amazon product co-purchasing network from March 12, 2003
amazon0505	410,236	2,439,437	Amazon product co-purchasing network from May 5, 2003
amazon0601	403,394	2,443,408	Amazon product co-purchasing network from June 1, 2003
cit-HepPh	34,546	420,877	Arxiv High Energy Physics paper citation network
cit-HepTh	2777	352,285	Arxiv High Energy Physics paper citation
email-EuAll	265,214	364,481	Email network from a EU research institution
p2p-Gnutella04	10,876	39,994	Gnutella peer to peer network from August 4, 2002
p2p-Gnutella24	26,518	65,369	Gnutella peer to peer network from August 24, 2002
p2p-Gnutella25	22,687	54,705	Gnutella peer to peer network from August 25, 2002
p2p-Gnutella30	36,682	88,328	Gnutella peer to peer network from August 30, 2002
p2p-Gnutella31	62,586	147,892	Gnutella peer to peer network from August 31, 2002
soc-Slashdot0811	7736	469,180	Slashdot social network from November 2008
soc-Slashdot0922	82,168	504,230	Slashdot social network from February 2009
soc-Epinions1	75,879	405,740	Who-trusts-whom network of Epinions.com
wiki-Vote	7115	100,762	Wikipedia who-votes-on-whom network
web-NotreDame	325,729	1,090,108	Web graph of Notre Dame
web-Stanford	281,903	1,992,636	Web graph of Stanford.edu
wiki-Talk	2,394,385	4,659,565	Wikipedia talk (communication) network
web-BerkStan	685,230	6,649,470	Web graph of Berkeley and Stanford
web-Google	875,713	4,322,051	Web graph from Google
cit-Patents	3,774,768	16,518,947	Citation network among US Patents

In general, note that the benchmark instances are assumed to be simple connected graphs without isolated vertices. For this purpose, we employed a preprocessing step for removing isolated vertices, self-loops and parallel edges.

5.3. Results and Discussion

The numerical results for the first two benchmark sets—-that is, for social networks—are presented in Table 5 which has the following structure. The first column indicates the name of the social network. For each of the fives greedy algorithms the table contains two columns. The first one, labeled *Val*, provides the objective function values of the solutions found for the corresponding problem instances. The other one, labeled *Time(s)*, shows the corresponding running times in seconds. In this context, note that a computation time of 0.0 means that the time was below 0.01 seconds. The best result per table row is highlighted in bold font. Moreover, note that provenly optimal results—see Table 6 for the full CPLEX results—are underlined.

Table 5. Numerical results for the small and large-scale real social networks
.

Network	Wang's greedy		Raei's greedy		Fei's greedy		Pan's greedy		IGA-PIDS	
	Val	Time(s)	Val	Time(s)	Val	Time(s)	Val	Time(s)	Val	Time(s)
CA-GrQc	2626	0.28	2623	0.30	2622	0.36	2612	0.0	**2607**	0.0
CA-HepTh	4598	0.95	4602	0.94	4582	1.17	4565	0.0	**4544**	0.0
CA-HepPh	4887	2.91	4886	2.91	4876	3.27	4857	0.015	**4817**	0.015
CA-AstroPh	7081	7.92	7085	7.91	7062	8.59	7030	0.031	**6953**	0.031
CA-CondMat	9869	7.58	9853	7.64	9837	8.11	9816	0.0	**9748**	0.015
Email-Enron	12,015	13.52	12,184	15.75	11,958	15.23	11,952	0.047	**11,843**	0.031
ncstrlwg2	3034	0.39	3025	0.39	3023	0.44	3026	0.0	**3010**	0.015
actors-data	3199	2.33	3205	2.41	3187	2.44	3215	0.015	**3174**	0.016
ego-facebook	1976	0.75	**1975**	0.84	**1975**	0.84	1978	0.062	**1975**	0.078
socfb-nips-ego	<u>**1398**</u>	0.02	<u>**1398**</u>	0.05	<u>**1398**</u>	0.05	<u>**1398**</u>	0.0	<u>**1398**</u>	0.016
socfb-Mich67	1481	0.56	1478	0.63	1473	0.55	1458	0.016	**1427**	0.015
socfb-Brandeis99	1535	0.97	1539	1.64	1529	0.98	1522	0.031	**1502**	0.032
soc-gplus	8341	1.97	**8247**	2.22	8267	2.56	8351	0.031	8289	0.031
Karate	31	0.0	31	0.0	<u>**30**</u>	0.0	32	0.0	31	0.0
Dolphins	<u>**15**</u>	0.0	<u>**15**</u>	0.0	<u>**15**</u>	0.0	<u>**15**</u>	0.0	<u>**15**</u>	0.0
Football	**68**	0.015	**68**	0.0	69	0.0	69	0.0	**68**	0.0
Jazz	**81**	0.015	82	0.0	**81**	0.0	83	0.0	**81**	0.0
Avg	3660.88	2.363	3664.47	2.565	3646.12	2.623	3645.82	0.015	**3616.59**	0.017

Table 6. Results before and after removing redundant vertices, and the comparison to CPLEX.

Network	Before		After		CPLEX	
	Val	Time(s)	Val	Time(s)	Val	Gap (%)
CA-GrQc	2610	0.0	**2607**	0.0	2587*	0.77
CA-HepTh	4559	0.0	**4544**	0.0	4471*	1.63
CA-HepPh	4855	0.0	**4817**	0.015	4718	2.1
CA-AstroPh	7034	0.031	**6953**	0.031	6740	3.16
CA-CondMat	9804	0.015	**9748**	0.015	9584	1.71
Email-Enron	11,914	0.047	**11,843**	0.031	11,682*	1.38
ncstrlwg2	3014	0.0	**3010**	0.015	2994*	0.53
actors-data	3214	0.015	**3174**	0.016	3092	2.65
ego-facebook	1977	0.062	**1975**	0.078	1973*	0.1
socfb-nips-ego	**<u>1398</u>**	0.015	**<u>1398</u>**	0.016	1398*	0
socfb-Mich67	1452	0.0	**1427**	0.015	1329	7.37
socfb-Brandeis99	1517	0.016	**1502**	0.032	1400	7.29
soc-gplus	8294	0.031	**8289**	0.031	8244*	0.55
Karate	**31**	0.0	**31**	0.0	30*	3.33
Dolphins	**<u>15</u>**	0.0	**<u>15</u>**	0.0	15*	0
Football	70	0.0	**68**	0.0	63*	7.93
Jazz	82	0.0	**81**	0.0	79*	2.53
Avg	3637.65	0.014	**3616.59**	0.017	3552.88	2.53

The experimental results allow us to make the following observations. IGA-PIDS is able to obtain the best solution for 15 out of 17 problem instances. In contrast, all competitors together only return the best solution in 7 out of 17 cases. In addition, we can observe that both Pan's greedy and IGA-PIDS are significantly faster than the other three competitors (approximately two orders of magnitude). Furthermore, IGA-PIDS is only marginally slower than Pan's greedy, which means that the improved solution quality obtained by IGA-PIDS is not achieved at the expense of a significantly increased computation time. Moreover, without taking our greedy approach into consideration, it is worth to note that Pan's greedy outperforms the other greedy algorithms both in term of solution quality and computational efficiency. Finally, while the performance of each of the four greedy algorithms starts to degrade in the context of the largest problem instances, this is not the case for IGA-PIDS.

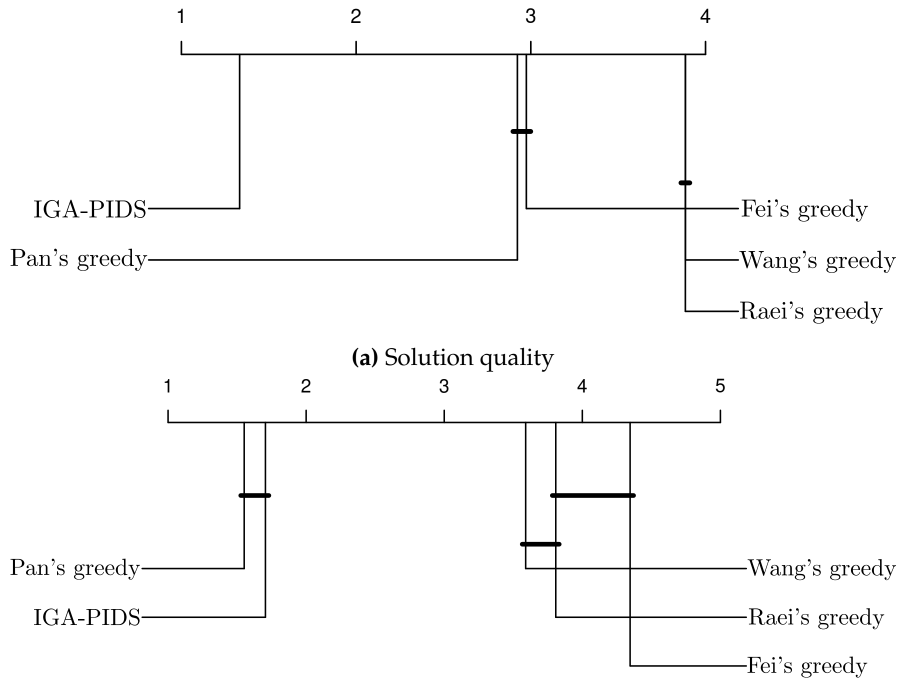
In order to study the effect of the proposed function to eliminate redundant vertices—see function Reduce($\cdot$) described in Section 4.2—Table 6 reports the performance of our greedy algorithm before (IGA-PIDS) and after (IGA-PIDS$^-$) removing redundant vertices. From these results it can be concluded that most solutions suffer from the existence of redundant vertices. Thus, removing them can provide more accurate results which comes, however, at the cost of additional computation time. Having said that, the increase of the average computation time is very moderate: from 0.014 seconds to only 0.017 seconds. In Table 6, we additionally present the results obtained from the ILP solver CPLEX. The obtained results show that the newest CPLEX version (12.10) is very efficient for the MPIDS problem. In fact, only in two cases (social networks socfb-nips-ego and Dolphins) our greedy algorithm is able to match the results of CPLEX. Moreover, CPLEX provides provenly optimal solutions in 11 out of 17 cases (within 2 hours of computation time). Provenly optimal results are marked by an asterisk. Nevertheless, the results provided by IGA-PIDS are, on average, only 2.53% worse than those of CPLEX (see last table column labeled Gap (%)).

Finally, the results for the large-scale SNAP networks are presented in Table 7, in the same format as outlined above. The results of CPLEX are also included. Moreover, note that no result is provided in those cases in which the computation time of the respective algorithm exceeded 7200 seconds (two hours). The results on these larger networks confirm our findings from the first two benchmark sets. However, the differences between the algorithms are now even more pronounced. Observe, for example, that—on average—IGA-PIDS produces solutions with more than 2000 vertices less per instance when compared to Pan's greedy. Moreover, the limits of CPLEX are clearly reached in the application to problem instances of the size and the complexity found in the Amazon* instances and cit-Patents. The results provided by CPLEX in these cases are far worse than those of IGA-PIDS, as indicated by the gaps. In fact, the results of IGA-PIDS improve by approx. 55% over the results of CPLEX in these cases. Therefore, IGA-PIDS is overall the best algorithm, even outperforming CPLEX with an average gap of approx. 42% between the IGA-PIDS solutions and the CPLEX solutions.

Table 7. Numerical results for the SNAP networks. Red color indicates those cases in which CPLEX fails to find good solutions.

Network	Wang's greedy		Raei's greedy		Fei's greedy		Pan's greedy		IGA-PIDS		CPLEX	
	Val	Time(s)	Val	Time(s)	Val	Time(s)	Val	Time(s)	Val	Time(s)	Val	Gap (%)
Amazon0302	136,448	1680.11	136,177	1565.20	135,502	1619.34	136,723	0.19	**134,569**	0.23	262,111	-48.66
Amazon0312	186,772	5862.28	188,194	5676.23	186,009	5777.75	183,108	0.56	**180,853**	0.67	400,727	-54.87
Amazon0505	189,392	6152.06	-	-	-	-	185,307	0.56	**183,114**	0.64	410,236	-55.63
Amazon0601	184,892	6833.42	186,126	6077.13	-	-	182,291	0.63	**179,964**	0.66	403,394	-55.39
Cit-HepPh	13,340	47.08	13,394	39.11	13,316	37.40	13,340	0.078	**13,111**	0.08	12,350	6.16
Cit-HepTh	11,549	24.81	11,671	25.31	11,531	25.77	11,544	0.078	**11,399**	0.08	10,740	6.14
Email-EuAll	106,178	521.58	**105,691**	676.42	105,815	1038.92	106,220	0.89	105,906	1.25	105,659*	0.23
p2p-Gnutella04	4310	1.41	4297	1.42	4294	1.68	4243	0.0	**4170**	0.0	3995	4.38
p2p-Gnutella24	8812	6.61	8794	6.63	8776	6.84	8750	0.015	**8665**	0.015	8457	2.46
p2p-Gnutella25	7682	4.69	7653	4.70	7659	4.86	7635	0.016	**7555**	0.0	7370	2.51
p2p-Gnutella30	12,321	16.41	12,314	16.27	12,285	17.19	12,254	0.016	**12,125**	0.015	11,859	2.24
p2p-Gnutella31	20,614	61.08	20,604	60.42	20,541	63.56	20,448	0.032	**20,268**	0.016	19,876*	1.97
Slashdot0811	19,115	98.34	19,567	101.80	19,126	137.48	18,571	0.047	**18,515**	0.032	18,419*	0.52
Slashdot0902	21,417	107.20	21,856	78.92	21,403	85.19	20,857	0.063	**20,782**	0.031	20,629*	0.74
soc-Epinions1	21,227	52.86	21,494	88.59	21,241	82.89	21,015	0.046	**20,986**	0.031	20,960*	0.12
Wiki-Vote	1570	0.75	1593	0.77	1564	0.83	1506	0.016	**1499**	0.015	1461*	2.60
web-NotreDame	144,654	1172.89	144,696	1223.31	144,391	1366.73	145,564	0.55	**144,385**	0.84	143,742	0.45
web-Stanford	139,970	2944.28	140,577	3635.30	139,812	3405.31	140,630	114.64	**139,346**	139.45	137,175	1.58
Wiki-Talk	-	-	-	-	-	-	499,392	46.125	**490,133**	42.13	480,063*	2.10
web-BerkStan	-	-	-	-	-	-	339,452	209.27	**337,388**	259.81	335,493	0.56
web-Google	-	-	-	-	-	-	396,836	3.49	**394,806**	3.67	389,079	1.47
cit-Patents	-	-	-	-	-	-	1,599,417	6.0	**1,576,091**	5.73	3,774,768	-57.63
Avg	-	-	-	-	-	-	184,322.86	17.42	**182,074.09**	20.70	317,207.41	-42.60

Finally, we decided to provide a statistical assessment of the obtained results by means of critical difference (CD) plots [31]. In order to produce the average rank of each greedy algorithm considering the complete set of 39 problem instances, the `scmamp` R package [31] first applies the Friedman test to compare the five approaches simultaneously. In this way, the hypothesis that the five techniques perform equally was rejected. Then, the package performs pairwise comparisons using the Nemenyi post-hoc test [32]. As mentioned above, the results are shown graphically by means of CD plots. Note that, in such a plot, each considered algorithm is placed on the horizontal axis according to its average ranking. The performances of those algorithms that are below the critical difference threshold (computed with a significance level of 0.05) are considered as statistically equivalent. This is indicated by bold horizontal bars joining the markers of the respective algorithm variants. Figure 2 shows two CD plots. The first one (Figure 2a) concerns the obtained solution quality. IGA-PIDS is clearly the best-performing algorithm, with statistical significance. Concerning computation times (Figure 2b) it can be observed that Pan's greedy is slightly faster than IGA-PIDS. However, no statistical difference can be detected between the running times of Pan's algorithm and those of IGA-PIDS.



(a) Solution quality

(b) Computation times

Figure 2. Critical difference plots

6. Conclusions

In this paper, we have studied an APX-hard combinatorial optimization problem in graphs and networks, the so-called minimum positive influence dominating set (MPIDS) problem. For tackling this problem, we present a simple and a fast improved greedy algorithm (IGA-PIDS) which is based on an effective exploitation of information provided by problem specific knowledge. The performance of IGA-PIDS is evaluated on 17 social networks of different sizes, ranging from 34 to 23628 vertices. Moreover, we have applied our own greedy algorithm, as well as the competitors from the literature, to large-scale complex networks from the SNAP data set. The previously existing greedy heuristics for the MPIDS problem were re-implemented by ourselves for this purpose. Numerical results show that our greedy algorithm is able to outperform the other greedy algorithms from the literature, providing solutions with a significantly higher quality, especially in the context of the larger SNAP networks. We were also able to show that CPLEX, even though performing very strongly on small and medium-sized problem instances, reaches its limits when tackling large-scale complex networks.

Concerning limitations of this work, it would certainly be interesting to relate network characteristics with problem difficulty. That is, in the future it would be very interesting to identify those network characteristics that make the problem difficult, both for CPLEX and for our greedy algorithm. Note that such characteristics are not necessarily the same for the two types of algorithms. Furthermore, the design of well-working metaheuristics for this problem seems very challenging. Considering the results of CPLEX from this work, it is clear that the two existing metaheuristics for the MPIDS problem can not compete with CPLEX for most of the 17 social networks that were tackled in earlier works. The design of novel metaheuristics for the MPIDS problem will benefit from our improved greedy heuristic. Moreover, given the results of CPLEX, it might be a good idea to develop hybrid algorithms that combine metaheuristic elements with those of exact solvers such as CPLEX.

Author Contributions: Methodology, S.B.; Programming, S.B.; Writing—original draft, S.B.; Writing—review and editing, S.B. and C.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by project CI-SUSTAIN funded by the Spanish Ministry of Science and Innovation (PID2019-104156GB-I00).

Acknowledgments: S. Bouamama is grateful to DGRSDT-MESRS (Algeria), for financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cai, S.; Hou, W.; Wang, Y.; Luo, C.; Lin, Q. Two-goal Local Search and Inference Rules for Minimum Dominating Set. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, Yokohama , Japan, 11–17 July 2020; pp. 1467–1473.
2. Li, J.; Potru, R.; Shahrokhi, F. A Performance Study of Some Approximation Algorithms for Computing a Small Dominating Set in a Graph. *Algorithms* **2020**, *13*, 339.
3. Li, R.; Hu, S.; Liu, H.; Li, R.; Ouyang, D.; Yin, M. Multi-Start Local Search Algorithm for the Minimum Connected Dominating Set Problems. *Mathematics* **2019**, *7*, 1173.
4. Yuan, F.; Li, C.; Gao, X.; Yin, M.; Wang, Y. A novel hybrid algorithm for minimum total dominating set problem. *Mathematics* **2019**, *7*, 222.
5. Zhou, Y.; Li, J.; Liu, Y.; Lv, S.; Lai, Y.; Wang, J. Improved Memetic Algorithm for Solving the Minimum Weight Vertex Independent Dominating Set. *Mathematics* **2020**, *8*, 1155.
6. Wang, F.; Camacho, E.; Xu, K. Positive influence dominating set in online social networks. In *International Conference on Combinatorial Optimization and Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 313–321.
7. Tankovska, H. Global social networks ranked by number of users 2021. Available online: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/ (accessed on 22 February 2021).
8. Fournier, A.K.; Hall, E.; Ricke, P.; Storey, B. Alcohol and the social network: Online social networking sites and college students' perceived drinking norms. *Psychol. Pop. Media Cult.* **2013**, *2*, 86.
9. Long, C.; Wong, R.C.W. Minimizing seed set for viral marketing. In proceedings of 2011 11th IEEE International Conference on Data Mining, Vancouver, Canada, 11–14 December 2011; pp. 427–436.
10. Günneç, D.; Raghavan, S.; Zhang, R. Least-cost influence maximization on social networks. *Informs J. Comput.* **2020**, *32*, 289–302.
11. Wang, G. Domination problems in social networks. PhD thesis, University of Southern Queensland, Queensland, Australia, 2014.
12. Rad, A.A.; Benyoucef, M. Towards detecting influential users in social networks. In *E-Technologies: Transformation in a Connected World*. Springer: Berlin/Heidelberg, Germany, 2011; pp. 227–240.
13. Wang, F.; Du, H.; Camacho, E.; Xu, K.; Lee, W.; Shi, Y.; Shan, S. On positive influence dominating sets in social networks. *Theor. Comput. Sci.* **2011**, *412*, 265–269.
14. Jungnickel, D. *Graphs, networks and algorithms*; Springer: Berlin/Heidelberg, Germany, 2005.
15. Raei, H.; Yazdani, N.; Asadpour, M. A new algorithm for positive influence dominating set in social networks. In proceedings of 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Istanbul, Turkey, 26-29 August 2012, pp. 253–257.
16. Fei, M.; Weidong, C. An improved algorithm for finding minimum positive influence dominating sets in social networks. *J. South China Norm. Univ.* **2016**, *48*, 59–63.
17. Pan, J.; Bu, T.M. A Fast Greedy Algorithm for Finding Minimum Positive Influence Dominating Sets in Social Networks. In proceedings of IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, Paris, France, 29 April–2 May 2019, pp. 360–364.
18. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Acm Comput. Surv.* **2003**, *35*, 268–308.
19. Hussain, K.; Salleh, M.N.M.; Cheng, S.; Shi, Y. Metaheuristic research: a comprehensive survey. *Artif. Intell. Rev.* **2019**, *52*, 2191–2233.
20. Lin, G.; Guan, J.; Feng, H. An ILP based memetic algorithm for finding minimum positive influence dominating sets in social networks. *Phys. Stat. Mech. Its Appl.* **2018**, *500*, 199–209.
21. Feo, T.A.; Resende, M.G. Greedy randomized adaptive search procedures. *J. Glob. Optim.* **1995**, *6*, 109–133.
22. Lin, G.; Luo, J.; Xu, H.; Xu, M. A Hybrid Swarm Intelligence-Based Algorithm for Finding Minimum Positive Influence Dominating Sets. In *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. Springer: Cham, Switzerland, 2019; pp. 506–511.
23. Bouamama, S.; Blum, C. A hybrid algorithmic model for the minimum weight dominating set problem. *Simul. Model. Pract. Theory* **2016**, *64*, 57–68.
24. Biggs, N.; Lloyd, E.K.; Wilson, R.J. *Graph Theory, 1736–1936*; Oxford University Press: Oxford, UK, 1986.
25. Girvan, M.; Newman, M.E. Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **2002**, *99*, 7821–7826.
26. Zachary, W.W. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **1977**, *33*, 452–473.
27. Lusseau, D.; Schneider, K.; Boisseau, O.J.; Haase, P.; Slooten, E.; Dawson, S.M. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behav. Ecol. Sociobiol.* **2003**, *54*, 396–405.
28. Gleiser, P.M.; Danon, L. Community structure in jazz. *Adv. Complex Syst.* **2003**, *6*, 565–573.
29. Rossi, R.A.; Ahmed, N.K. The Network Data Repository with Interactive Graph Analytics and Visualization. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.

30. Leskovec, J.; Krevl, A. SNAP Datasets : Stanford Large Network Dataset Collection, 2014. Available online: https://snap.stanford.edu/data/ (accessed on 26 February 2021).
31. Calvo, B.; Santafé, G. scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. *R. J.* **2016**, *8*, 248–256.
32. García, S.; Herrera, F. An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. *J. Mach. Learn. Res.* **2008**, *9*, 2677 – 2694.