symmetry

Article

Solving Robust Weighted Independent Set Problems on Trees and under Interval Uncertainty

Ana Klobučar [1,*] **and Robert Manger** [2]

1 Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, 10000 Zagreb, Croatia
2 Department of Mathematics, Faculty of Science, University of Zagreb, 10000 Zagreb, Croatia; manger@math.hr
* Correspondence: aklobucar@fsb.hr

Abstract: The maximum weighted independent set (MWIS) problem is important since it occurs in various applications, such as facility location, selection of non-overlapping time slots, labeling of digital maps, etc. However, in real-life situations, input parameters within those models are often loosely defined or subject to change. For such reasons, this paper studies robust variants of the MWIS problem. The study is restricted to cases where the involved graph is a tree. Uncertainty of vertex weights is represented by intervals. First, it is observed that the max–min variant of the problem can be solved in linear time. Next, as the most important original contribution, it is proved that the min–max regret variant is NP-hard. Finally, two mutually related approximation algorithms for the min–max regret variant are proposed. The first of them is already known, but adjusted to the considered situation, while the second one is completely new. Both algorithms are analyzed and evaluated experimentally.

Keywords: weighted graph; independent set; robust optimization; interval uncertainty; tree; complexity; approximation algorithm

check for updates

Citation: Klobučar, A.; Manger, R. Solving Robust Weighted Independent Set Problems on Trees and under Interval Uncertainty. *Symmetry* **2021**, *13*, 2259. https://doi.org/10.3390/sym13122259

Academic Editor: Jian-Qiang Wang

Received: 22 October 2021
Accepted: 20 November 2021
Published: 27 November 2021

1. Introduction

Our introduction starts with some well-known definitions, which can be found in many textbooks, e.g., [1,2]. Let us consider an undirected graph G. Obviously, G is a symmetric object, since its edges are bidirectional. Or differently speaking, the adjacency matrix of G is symmetric [1]. An *independent set* is a set of vertices of G that are not adjacent to each other. Suppose that vertices of G are given non-negative weights. Then, the weight of an independent set is defined as the sum of its vertex weights. The (conventional) maximum weighted independent set (MWIS) problem consists of finding an independent set whose weight is as large as possible.

Our work is concerned with *robust optimization* [3–7]. We consider robust variants of the MWIS problem, where vertex weights are uncertain [8–12]. Thereby, uncertainty is expressed through a finite set of scenarios. Each scenario gives one possible combination of vertex weights. It is assumed that all weights are non-negative integers. The set of scenarios can be given explicitly—then we speak about *discrete uncertainty*. Or possible weights of a vertex can be given by an interval—then the set of all scenarios is implicitly given as the Cartesian product of all intervals, and we speak about *interval uncertainty*.

No matter which uncertainty representation is used, there are still several possibilities how to choose a robust solution of a problem. Each possibility corresponds to a different *criterion of robustness*. In this paper, two robust variants of the MWIS problem are considered, which correspond to the two most popular robustness criteria. The first criterion is called max–min [3] or absolute robustness [7]—it selects the independent set whose minimal weight over all scenarios is as large as possible. The second criterion is called min–max regret [3] or robust deviation [7]—it selects the independent set whose maximal deviation of weight from the conventional optimum, measured over all scenarios, is as small as possible.

Although the considered robust MWIS problem variants seem to be quite abstract, they actually occur in many applications. Thereby uncertainty in vertex weights corresponds to the fact that in real-life situations, many input parameters are loosely defined or subject to change. The most common application is facility location [13], where positions of facilities (e.g., shopping malls, gas stations) need to be chosen, so that some kind of profit is maximized and the facilities are not too close one to another. Another interesting application is selection of program slots of television channels for giving an advertisement [14]. To maximize the effect of advertising, it is important that the slots do not overlap in time and that they attract as many viewers as possible. A relatively new application is labeling of digital maps [10,15]. Such maps must display labels that correspond to certain points of interest (restaurants, cinemas, etc.). The shown labels should be chosen according to personal preferences of the viewer, and they should not overlap on the display.

This paper is concerned with solving the mentioned two robust MWIS problem variants on a special type of graphs, i.e., on *undirected trees* (or *trees* for short). Thereby, it is assumed that uncertainty in vertex weights is represented by intervals. Our goal is to study computational complexity of the considered robust variants, and also to propose suitable exact or approximate algorithms for their solution.

Now, we will explain our motivation for considering trees. Since the conventional MWIS problem is NP-hard in general [16], it is obvious that all its robust variants must also be NP-hard in general. But on the other hand, it is known that the conventional MWIS problem can be solved in polynomial (even linear) time if the involved graph is a tree [17]. This leaves a possibility that robust variants of the same problem can also be solved more efficiently on trees than on general graphs. Thus an interesting topic for research would be to explore computational complexity of robust MWIS problem variants on trees. Or in other words, it would be interesting to see whether restriction to trees assures the same computational advantage in the robust setting as in the conventional setting. Additional motivation for considering trees comes from the fact that such graphs occur in certain applications, e.g., some forms of facility location.

Our work is related to some other works from the literature. We have been inspired by [8,12], where an analogous complexity analysis of robust MWIS problem variants has been done for so-called interval graphs rather than for trees. Interval graphs are another class of graphs where the conventional MWIS problem can be solved in polynomial time [14,18]. It is important to note that interval graphs and trees are two different classes, i.e., none of them is a subset of the other [11]. Thus the available results from [8,12] regarding interval graphs cannot be applied to trees, and a separate study of complexity is needed.

This work is also in a close relationship with our previous paper [11], in which the same robust MWIS problem variants on trees have been considered under discrete uncertainty representation. More precisely, this paper is in fact a sequel of [11], where the results from [11] are extended to interval uncertainty representation. Thereby a seemingly small difference, dealing with uncertainty representation, produces radically different results. Indeed, the max–min problem variant considered in [11] is NP-hard, while in this paper it is solvable in linear time. The min–max regret variant turns out to be NP-hard in both papers, but the corresponding proofs of NP-hardness are quite different. The algorithms proposed in the two papers do not resemble one to another since they assume different sets of scenarios.

Apart from this introduction, the rest of the paper is organized as follows. Section 2 is concerned with solving the max–min variant of the MWIS problem on trees under interval uncertainty. Section 3 studies complexity of the corresponding min–max regret variant. Section 4 presents a simple approximation algorithm for the min–max regret variant, and analyzes its speed and accuracy when it is applied to our problem instances. Section 5 constructs an extended version of the same algorithm aimed to achieve better accuracy. Section 6 reports on an experimental evaluation of both algorithms within our setting. The final Section 7 gives conclusions.

2. Max–Min Variant

As already announced, we study robust variants of the MWIS problem. Thus, we consider independent sets in a graph G whose vertices are assigned non-negative integer weights. Uncertainty in weights is expressed through a finite set of scenarios S. More precisely, let n be the number of vertices in G. Denote the vertices with $v_1, v_2, \ldots, v_n$. Then, each scenario $s \in S$ is specified by a list of integers $w_1^s, w_2^s, \ldots, w_n^s$, where w_i^s is the weight of v_i under s.

Let us denote the weight of an independent set X under scenario s by $F(X, s)$. Obviously, it holds that $F(X, s) = \sum_{v_i \in X} w_i^s$. In this section, we restrict it to the max–min variant of the problem, which is defined in terms of $F(X, s)$ as follows.

Definition 1. *A max–min solution to the MWIS problem is an independent set X_A that maximizes the function $\min_{s \in S} F(X, s)$ over the whole collection of possible independent sets.*

From now on, it is assumed that uncertainty in vertex weights is expressed by intervals. It means that the weight of v_i can take any integer value from a given integer interval $[l_i, u_i]$. The set of scenarios S is then implicitly given as the Cartesian product of all intervals. For any scenario $s \in S$ it holds that $w_i^s \in [l_i, u_i]$.

Solving the max–min variant of the MWIS problem under interval uncertainty is closely related to the so-called *minimum scenario*. It is the scenario s where each vertex v_i has the minimum possible weight, i.e., $w_i^s = l_i$. Namely, the following well-know claim is valid.

Proposition 1. *Under interval uncertainty, a max–min solution to the MWIS problem is obtained by solving the conventional (non-robust) MWIS problem according to the minimum scenario.*

The proof of Proposition 1 can be found in [12]. The same claim in a more general setting is also stated without proof in [3]. According to Proposition 1, the max–min variant of the MWIS problem under interval uncertainty is as hard as the conventional variant. However, let us now assume additionally that our graph G is a tree. Then, according to [17], the conventional variant can be solved in linear time. Thus, the following consequence of Proposition 1 and [17] holds.

Proposition 2. *With restriction to trees and under interval uncertainty, a max–min solution to the MWIS problem can be obtained in time $\mathcal{O}(n)$. Here, n is the number of vertices in the involved tree.*

Proof. The conventional problem instance based on the minimum scenario specified by Proposition 1 can be constructed in time $\mathcal{O}(n)$. The constructed instance can be solved by the algorithm from [17] again in time $\mathcal{O}(n)$. Thus, the total time is $\mathcal{O}(n)$. □

Thus, in our considered situation (restriction to trees, interval uncertainty), the max–min variant of the MWIS problem is easy to solve. Indeed, a very fast exact algorithm is available. There is no need to design new algorithms. Therefore the max–min variant will be skipped from the rest of the paper.

3. Min–Max Regret Variant

As in the previous section, we again study a robust variant of the MWIS problem. Thus, we again consider independent sets in a graph G whose vertices are assigned non-negative integer weights. Uncertainty in weights is still expressed through a finite set of scenarios S. However, now we focus on the min–max regret variant of the problem.

Along with the notation from the previous section, some additional symbols will also be used. Indeed, $F^*(s)$ will denote the optimal solution weight for the conventional MWIS problem instance with vertex weights set according to scenario $s \in S$. For an independent set X and a scenario s, $D(X, s)$ will be the "regret" (deviation from optimum) of X under s, i.e., $D(X, s) = F^*(s) - F(X, s)$. For an independent set X, $R(X)$ will denote the maximal regret of X, i.e., $R(X) = \max_{s \in S} D(X,s)$. With this notation, the min–max regret variant is defined as follows.

Definition 2. *A min–max regret solution to the MWIS problem is an independent set X_D that minimizes the function $R(X) = \max_{s \in S} D(X,s) = \max_{s \in S}(F^*(s) - F(X,s))$ over the whole collection of possible independent sets.*

From now on, it will again be assumed that uncertainty in vertex weights is expressed by intervals. Thus, again, the weight of a vertex v_i can take any integer value from a given integer interval $[l_i, u_i]$, and the set of scenarios is the Cartesian product of all those intervals. For any independent set X, we can construct the corresponding "worst" scenario s_X, where the weight of v_i is equal to l_i if $v_i \in X$, and equal to u_i if $v_i \notin X$. It is interesting to note that the maximal regret of X can be characterized in terms of its worst scenario s_X in the following way.

Proposition 3. *Within the MWIS problem under interval uncertainty, the maximal regret of an independent set X is achieved on its worst scenario s_X, i.e.,*

$$R(X) = D(X, s_X) = F^*(s_X) - F(X, s_X).$$

Proof. In [3], the same claim has been proved in a more general setting, but for the case where the associated conventional (non robust) problem is a minimization problem. That proof will now be adjusted for maximization, as it is needed for our MWIS problem. Indeed, let X be an independent set, $s \in S$ a scenario, s_X the worst scenario for X, X_s^* the optimal solution for scenario s, and $X_{s_X}^*$ the optimal solution for scenario s_X. Then, it holds:

$$\begin{aligned}
D(X,s) &= F^*(s) - F(X,s) \\
&= \sum_{v_i \in X_s^* \setminus X} w_i^s - \sum_{v_i \in X \setminus X_s^*} w_i^s \quad \text{(since the common summands cancel)} \\
&\le \sum_{v_i \in X_s^* \setminus X} u_i - \sum_{v_i \in X \setminus X_s^*} l_i \quad (\text{since } l_i \le w_i^s \le u_i) \\
&= F(X_s^*, s_X) - F(X, s_X) \quad \text{(again, since the common summands cancel)} \\
&\le F(X_{s_X}^*, s_X) - F(X, s_X) \quad (\text{since } X_{s_X}^* \text{ is optimal for } s_X) \\
&= F^*(s_X) - F(X, s_X) \;=\; D(X, s_X).
\end{aligned}$$

Thus, $D(X,s) \le D(X,s_X)$ for any s, and therefore $R(X) = D(X,s_X)$. □

As in the previous section, we are again interested in a special situation under interval uncertainty, where the graph G involved in the MWIS problem is a tree. Since in such situation both the conventional and the max–min variant of the problem can be solved in linear time, one can hope that the min–max regret variant can also be solved with the same efficiency. Unfortunately, this is not true. Namely, the following theorem holds.

Theorem 1. *With restriction to trees and under interval uncertainty, finding a min–max regret solution to the MWIS problem is NP-hard.*

Proof. In [8], there is a similar theorem that refers to the so-called interval graphs instead of trees. Our proof is analogous to the proof from [8], but adjusted for trees. It is based on polynomial reduction of the NP-complete *partition problem* [16] to our robust problem. Here is the definition of partition:

Instance: a list of positive integers $a_1, a_2, \ldots, a_n$.
Question: is there a subset of indices $I \subset \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Our polynomial reduction works as follows. For a given instance of the partition problem we compute $b = \frac{1}{2}\sum_{i=1}^{n} a_i$ and construct the corresponding instance of the MWIS problem shown by Figure 1. The tree involved in our instance consists of the root v_0 and of n vertical segments. In the i-th segment there are two vertices: the "upper" one v_{i1} and the "lower" one v_{i2}. Uncertainty of vertex weights is expressed through intervals. More precisely:

- The weight of v_0 is between 0 and b.
- For any $i = 1, 2, \ldots, n$, the weight of v_{i1} is between $3b - \frac{3}{2}a_i$ and $3b$.
- For any $i = 1, 2, \ldots, n$, the weight of v_{i2} is exactly $3b - a_i$.

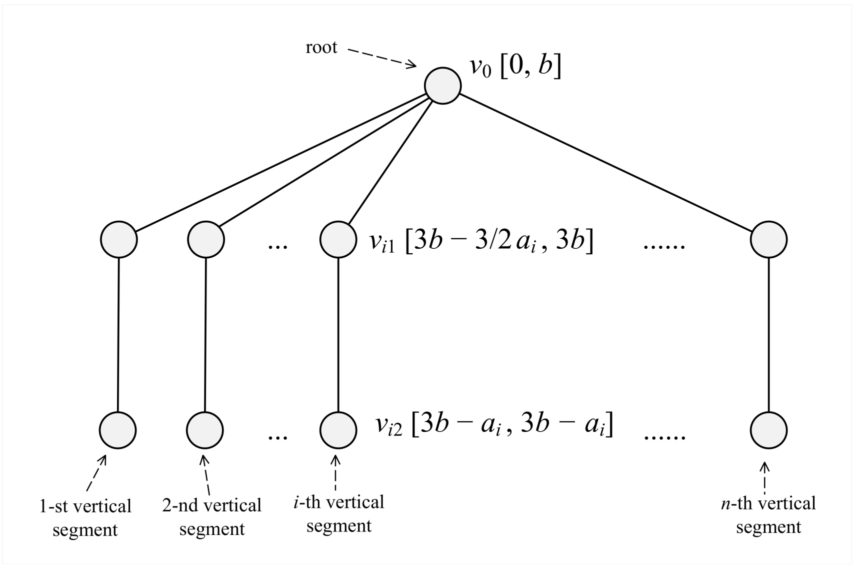

Figure 1. A tree that reduces a partition problem instance to a MWIS problem instance.

In order to find the solution of our MWIS problem instance, one can restrict to *nontrivial* (non-expandable) independent sets (i.e., those that cannot be expanded with an additional vertex). Obviously, there exist two types of nontrivial independent sets:

Type 1: The root v_0 is included. For each $i = 1, 2, \ldots, n$ the lower vertex v_{i2} is included.

Type 2: The root v_0 is not included. For each $i = 1, 2, \ldots, n$ one among vertices v_{i1} and v_{i2} is included (but not both). For at least one i, the upper vertex v_{i1} is included.

Let us denote with opt_D the maximal regret of a min–max regret solution X_D to our constructed MWIS problem instance, i.e., $opt_D = R(X_D) = \min R(X)$ over all independent sets X of type 1 or 2. We claim the following:

The answer to the given partition problem instance is "yes" if and only if $opt_D \leq \frac{3}{2}b$.

We prove the above claim by proving both directions.

The "if" direction. Suppose that the answer to the partition problem instance is "yes". Then, there exists a set of indices $I \subset \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i = b$. We construct the independent set X by excluding v_0, and by choosing v_{i1} for $i \in I$ and v_{i2} for $i \notin I$, respectively. The worst scenario s_X then looks as follows:

- v_0 has weight b,
- v_{i1} has weight $3b - \frac{3}{2}a_i$ if $i \in I$, and $3b$ if $i \notin I$,
- v_{i2} has weight $3b - a_i$ if $i \in I$, and again $3b - a_i$ if $i \notin I$.

We compute $F(X, s_X)$:

$$\begin{aligned} F(X, s_X) &= \sum_{i \in I}(3b - \frac{3}{2}a_i) + \sum_{i \notin I}(3b - a_i) \\ &= 3nb - \frac{5}{2}b. \end{aligned}$$

We compute $F^*(s_X)$, i.e., we find the conventional optimum under scenario s_X by considering different solutions and identifying the best of them.

- Type 1 solution has the weight

$$b + \sum_{i=1}^{n}(3b - a_i) = 3nb - b.$$

- The best type 2 solution is obtained so that in each vertical segment of T the heavier among two vertices is chosen. The obtained weight is

$$\sum_{i \in I}(3b - a_i) + \sum_{i \notin I} 3b = 3nb - b.$$

Thus,

$$F^*(s_X) = \max\{3nb - b, 3nb - b\} = 3nb - b.$$

Next, we apply Proposition 3 to compute the maximal regret for X:

$$\begin{aligned} R(X) &= F^*(s_X) - F(X, s_X) \\ &= 3nb - b - 3nb + \frac{5}{2}b \\ &= \frac{3}{2}b. \end{aligned}$$

Since opt_D is the minimum over all maximal regrets, it holds that

$$opt_D \leq R(X) \leq \frac{3}{2}b.$$

The "only if" direction. Suppose that $opt_D \leq \frac{3}{2}b$. Then, there exists an independent set X such that $R(X) \leq \frac{3}{2}b$. Thereby X cannot be of type 1, namely it can easily be checked that for the type 1 solution the maximal regret is equal to $2b$, which is surely $> \frac{3}{2}b$. Thus, our X must be of type 2, i.e., it does not include v_0, for any i it contains either v_{i1} or v_{i2} (but not both), and for at least one i it contains v_{i1}.

Let I' be the subset of $\{1, 2, \ldots, n\}$ such that $i \in I'$ if $v_{i1} \in X$, and $i \notin I'$ if $v_{i2} \in X$. The worst scenario s_X looks as follows:

- v_0 has weight b,
- v_{i1} has weight $3b - \frac{3}{2}a_i$ if $i \in I'$, and $3b$ if $i \notin I'$,
- v_{i2} has weight $3b - a_i$ if $i \in I'$, and again $3b - a_i$ if $i \notin I'$.

We compute $F(X, s_X)$:

$$\begin{aligned} F(X, s_X) &= \sum_{i \in I'}(3b - \frac{3}{2}a_i) + \sum_{i \notin I'}(3b - a_i) \\ &= 3nb - 2b - \frac{1}{2}\sum_{i \in I'} a_i. \end{aligned}$$

We compute $F^*(s_X)$. Thus, we search the conventional optimum under scenario s_X by comparing all feasible solutions.

- Type 1 solution has the weight

$$b + \sum_{i=1}^{n}(3b - a_i) = 3nb - b.$$

- The "heaviest" type 2 solution is obtained so that in each vertical segment of T the heavier among two vertices is chosen. The obtained weight is

$$\sum_{i\in I'}(3b-a_i)+\sum_{i\notin I'}3b=3nb-\sum_{i\in I'}a_i.$$

Thus,

$$F^*(s_X)=\max\left\{3nb-b,\ 3nb-\sum_{i\in I'}a_i\right\}.$$

Now, we can compute the maximal regret for X by using Proposition 3:

$$\begin{aligned}R(X)&=F^*(s_X)-F(X,s_X)\\&=\max\left\{b+\frac{1}{2}\sum_{i\in I'}a_i,\ 2b-\frac{1}{2}\sum_{i\in I'}a_i\right\}.\end{aligned}$$

From the fact that $R(X)\le\frac{3}{2}b$, one can conclude the following:

- It cannot be true that $\sum_{i\in I'}a_i>b$ since otherwise the left-hand part within the above $\max\{\ldots,\ldots\}$ would be $>\frac{3}{2}b$.
- It cannot be true that $\sum_{i\in I'}a_i<b$ since otherwise the right-hand part within the above $\max\{\ldots,\ldots\}$ would be $>\frac{3}{2}b$.

Thus, the exact equality must hold, i.e., $\sum_{i\in I'}a_i=b$, which means that I' determines a partition. □

4. Simple Algorithm

In the previous section, it has been proved that, under interval uncertainty, the min–max regret variant of the MWIS problem is NP hard, even with restriction to trees. Thus, unless P = NP, one cannot hope to find an efficient algorithm that solves the considered problem variant under the considered circumstances. The only feasible solution strategy is trying to find a good approximate algorithm.

In this section, we propose a simple approximate algorithm based on the so-called *average scenario*. It is the scenario where each vertex v_i has the average weight from its interval $[l_i,u_i]$, i.e., the (possibly non-integer) weight $(l_i+u_i)/2$. Going into more details, the proposed algorithm works as follows.

- For a given instance of the considered MWIS problem variant, find the average scenario.
- Solve the conventional (non-robust) MWIS problem instance according to the average scenario.
- The obtained solution (independent set) $\bar{X}$ serves as an approximation of a min–max regret solution.

The considered algorithm is not only an approximate, but also an *approximation algorithm* [19], in the sense that it allows a constant approximation ratio. Allowing a constant approximation ratio q means in our case the following. If $\bar{X}$ is an independent set produced by the algorithm and opt_D is the maximal regret of a corresponding truly optimal independent set, then $R(\bar{X})\le q\cdot opt_D$. Thereby the same q is used for all problem instances.

The described algorithm is in fact well known in the literature [3,8]. It can be applied in more general situations, but we believe that it is especially advantageous in our situation. Namely, its properties are then summarized by the following proposition.

Proposition 4. *With restriction to trees and under interval uncertainty, the algorithm based on the average scenario finds an approximate min–max regret solution to the MWIS problem in time $\mathcal{O}(n)$. Here, n is the number of vertices in the involved tree. The algorithm achieves the approximation ratio 2, and this ratio cannot be improved for trees.*

Proof. In [8], it has been proved that the considered algorithm is an approximation algorithm with a constant approximation ratio not greater than 2. The proof from [8] is valid for general graphs, and can therefore be applied to trees. Our estimate for computing time is specific for trees, and it comes from the fact that the associate conventional problem (with the average scenario) can be solved by the linear-time algorithm from [17]. Note that the algorithm from [17] also works correctly when vertex weights are non-negative real numbers. To complete the proof, one must demonstrate that the general approximation ratio 2 from [8] cannot be improved for trees. We do this by presenting our own original problem instance involving a tree, where the algorithm returns a solution $\bar{X}$ such that $R(\bar{X})$ is exactly equal to $2 \cdot opt_D$. Indeed, our problem instance is specified by Figure 2. The figure shows a tree consisting of five vertices, each with a given interval for its weight. □

Now, the example from Figure 2 will be explained in more detail. The vertices of the shown tree are denoted with v_0, v_1, v_2, v_3 and v_4. The intervals for their weights are in turn $[12, 24]$, $[58, 84]$, $[84, 85]$, $[31, 47]$, and $[72, 97]$. It is easy to check that there are only four nontrivial (non-expandable) independent sets, i.e., $\{v_0, v_3, v_4\}$, $\{v_1, v_2\}$, $\{v_1, v_4\}$, and $\{v_2, v_3\}$. The weights of the five vertices according to the average scenario are in turn 18, 71, 84.5, 39, and 84.5. The weights of the four independent sets under the average scenario are in turn 123.5, 155.5, 155.5, and 123.5. Thus, our approximation algorithm can return the independent set $\bar{X} = \{v_1, v_2\}$ as its approximate solution. On the other hand, it can be checked by straightforward computation that the maximal regrets for the four independent sets are in turn 54, 26, 13, and 66. Thus, the maximal regret of our approximate solution is $R(\bar{X}) = 26$, while the minimal maximal regret over all solutions is $opt_D = 13$. Thus, it really holds that $R(\bar{X}) = 2 \cdot opt_D$.

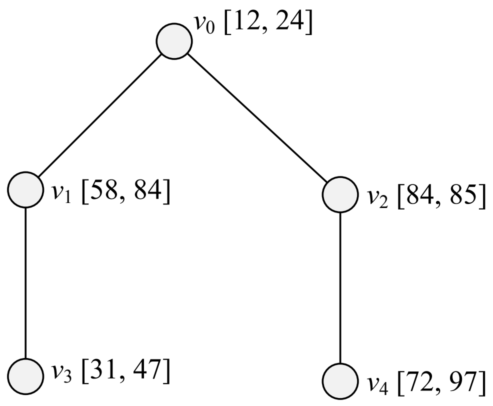

Figure 2. A MWIS problem instance where the maximal regret of the approximate solution is 2 times larger than the minimal maximal regret.

5. Extended Algorithm

In this section, we present an extended algorithm for solving the min–max regret variant of the MWIS problem on trees and under interval uncertainty. It is built upon the simple algorithm from the previous section. It starts with the solution obtained by the simple algorithm and tries to improve that solution through local search. Obviously, the solution obtained with the extended algorithm is never worse than the one obtained with the simple algorithm. Consequently, the extended algorithm can also be regarded as an approximation algorithm with the same or better approximation ratio.

The outline of our extended algorithm is described by the pseudo-code from Figure 3. The code obeys the standard local-search strategy developed, e.g., in [20,21]. Thus, for a given robust MWIS problem instance the algorithm first finds an *initial solution*, i.e., a feasible independent set. In our case the initial solution is constructed by invoking the simple algorithm from the previous section. Next, the initial solution is repeatedly improved, thus producing a series of *current solutions*. In each iteration, the algorithm generates a *neighborhood*, which consists of various feasible perturbations of the current solution.

All independent sets in the neighborhood are evaluated according to the maximal regret function $R(\,)$. The best member of the neighborhood (i.e., the one whose maximal regret is minimal) is identified. If that member is better than the current solution (i.e., its maximal regret is smaller), it becomes the new current solution and the algorithm resumes with a new iteration. Otherwise, the algorithm stops. The last current solution is regarded as nearly optimal (according to the min–max regret criterion).

```
Local search for the min-max regret variant of the
MWIS problem on trees and under interval uncertainty {
   input the problem instance;
   $X$ = the independent set computed by the simple algorithm
   from Section 4 applied to the given problem instance;
   evaluate $X$ according to the maximal regret function $R(\,)$;
   while (true) {
       generate the neighborhood $\mathcal{N}$ of $X$ consisting
       of a desired number of feasible independent sets;
       evaluate all independent sets in $\mathcal{N}$
       according to the maximal regret function $R(\,)$;
       $Y$ = the best-evaluated independent set in $\mathcal{N}$;
       if ($Y$ is better than $X$)
           $X$ = $Y$;
       else
           break;
   }
   output $X$ as the min-max regret solution;
}
```

Figure 3. Pseudo-code of the extended algorithm.

A special property of our extended algorithm is that it considers only such solutions (independent sets) that are optimal (in the conventional sense) for some scenario. That scenario can consist either of integer or of real vertex weights. The mentioned property holds not only for the initial solution (which is optimal for the average scenario), but also for all subsequently generated current solutions and members of their neighborhoods. Consequently, each considered solution can be identified with its scenario. Modification of a current solution within local search is accomplished by perturbing the respective scenario rather than perturbing the independent set itself. Indeed, it is much easier to perturb the scenario in a feasible way. Transformation of a scenario into the corresponding independent set is accomplished by the linear-time algorithm from [17].

Now, it will be explained in more detail how the solutions considered by our extended algorithm are evaluated and compared.

- Let σ be the current scenario (equivalent of the current solution). We compute the respective independent set X by applying the algorithm from [17]. In order to evaluate X, we first construct its "worst" scenario s_X (see Section 3), and then compute its maximal regret $R(X)$ according to the formula from Proposition 3:

$$R(X) = F^*(s_X) - F(X, s_X).$$

  Note that $F^*(s_X)$ in the above formula requires an additional call of the algorithm from [17].
- Suppose that the current scenario σ has been perturbed in some way. Let τ be the obtained perturbed version of σ. Similarly as before, we switch from τ to its

corresponding independent set Y by using the algorithm from [17]. To evaluate Y, we again construct its "worst" scenario s_Y and compute its maximal regret

$$R(Y) = F^*(s_Y) - F(Y, s_Y).$$

Thereby the value $F^*(s_Y)$ is obtained by calling once more the algorithm from [17].

Within one iteration of local search, σ usually does not need to be evaluated since its respective maximal regret $R(X)$ is already available from the previous iteration. However, construction and evaluation of τ must be repeated many times in order to produce a neighborhood with a desired size. Among all instances of τ, one is identified whose respective Y has the smallest maximal regret $R(Y)$. As explained earlier, the identified τ will become the next current scenario if its $R(Y)$ is smaller than $R(X)$.

Next, we have to explain how a current scenario σ is transformed into one of its perturbed versions τ. Perturbation can be done in many ways. In the present variant of the algorithm, two parameters with values between 0 and 1 are chosen in advance:

- Perturbation probability π,
- Perturbation intensity δ.

Then, for each vertex in the tree the probability that its weight will change during perturbation is equal to π. The increase or decrease of weight (when it happens) is limited to the respective weight-interval width multiplied by the factor δ. Of course, change is done only to the extent that would not violate the original interval restriction for that particular weight. For instance, let the number of vertices be $n = 500$ and $\pi = 0.2$. Let vertex weights be in the range between 1 and 50 and $\delta = 0.1$. Then, each of 500 vertices perturbs its weight independently one from another with probability 20%. Whenever a weight changes, the amount of change is a random number between $\pm$10% of 50 (i.e., between -5 and +5).

Let us say something about computational complexity of the extended algorithm. Construction of an instance of τ is done in time linear with respect to the tree size n. Evaluation of σ or of any instance of τ requires two invocations of the (linear-time) algorithm from [17]. Consequently, for a constant neighborhood size, the computing time of an iteration is linear with respect to n.

Regarding space complexity, the extended algorithm does not require a lot of memory. It only needs to store information about the tree and about the current best solution. Iterations do not require new space because a new better solution replaces the current best solution. Consequently, the total space requirement is linear with respect to the tree size n.

6. Experimental Evaluation

In this section, we present our experimental evaluation of the two algorithms from Sections 4 and 5, respectively. It would be nice if the algorithms could be tested on some established benchmark instances for robust variants of the MWIS problem. However, to the best of our knowledge, such benchmarks do not exist, especially not on trees or with interval uncertainty. Therefore, we have generated our own three test groups, the first comprising 30, the second 120, and the third again 120 robust MWIS problem instances. We call the three groups *small*, *medium-sized* and *large* instances, respectively.

Here follows a more detailed description of our test data. Of course, all instances in our test groups are posed on trees, and uncertainty of their vertex weights is expressed by intervals. Additionally, all instances are meant to be solved according to the min–max regret criterion. Some parameters, such as numbers of vertices, bounds for tree outspread or bounds for vertex weights, are chosen in advance. The remaining details are generated randomly. Indeed:

- Small instances are based on random trees consisting of $n = 20$ vertices, where each vertex has at most 3 children. The lower bound for a vertex weight is always 1, and the upper bound is at most 5.
- Medium-sized instances comprise random trees with $n = 500$ vertices. There are four subgroups of 30 instances, characterized by different outspread bounds. More

precisely, in the first subgroup, each vertex in a tree can have at most three children, while the maximum number of children in the second, third and fourth subgroup is 5, 10 and 15, respectively. Regardless of a subgroup, the range for a vertex weight is from 1 to at most 50.

- Large instances contain random trees having n = 10000 vertices. Again, there are four subgroups of 30 instances, characterized by different outspread bounds. Depending on a subgroup, a vertex can have at most 3, 5, 10 and 15 children, respectively. In all four subgroups, vertex weights are in the interval from 1 to at most 1000.

A full specification of all problem instances can be found in our repository at the address http://hrzz-rodiopt.math.pmf.unizg.hr (accesed on 20 November 2021).

The instances from the first group may look rather small (only 20 vertices). However, according to our experience, they are in fact the largest ones that can be solved to optimality by using a straightforward mathematical model and a general-purpose software such as CPLEX [22]. Indeed, our straightforward model is based on constraints corresponding to extremal scenarios [12]. For 20 vertices, there are as many as $2^{20} = 1{,}048{,}576$ such constraints. All of them are explicitly listed in the model, thus reaching the limits of CPLEX capabilities. It is true that some larger problem instances could still be solved exactly by more sophisticated models and algorithms [3], where only a small part of the constraint set is initially used and then gradually extended. However, exploration of such methods is beyond the scope of this paper.

The simple algorithm from Section 4, as well as the extended algorithm from Section 5, have been implemented in the Java programming language. Both implementations also comprise the linear-time algorithm from [17] as a subroutine. In the second implementation, the two parameters π and δ (see Section 5), as well as the neighborhood size for local search, can freely be chosen and specified as input data. To measure accuracy of our approximation algorithms (at least on small instances), we have also used the previously mentioned CPLEX package. All software has been installed on the same computer, with an Intel Core I7-975OH @2.60 GHz processor and 16 GB of RAM, running a 64-bit operating system. Our original source code is available in the same repository as the problem instances.

In the experiments we have solved each problem instance from each of the three test groups by both approximation algorithms. The small instances (first test group) have additionally been solved to optimality by CPLEX (with the straightforward mathematical model and explicitly listed constraints). The medium-sized and large instances (second and third test group) have repeatedly been solved by the extended algorithm, each time with a different combination of parameters π and δ. In all experiments, the neighborhood size within local search has been fixed to 100.

For each solution we have recorded the obtained maximal regret $R()$ and the corresponding computing time. The results of the experiments are summarized in Tables 1–3. Thereby Table 1 corresponds to small, Table 2 to medium-sized, and Table 3 to large instances.

Let us now take a closer look at Table 1. Each row of the table corresponds to one of the 30 small problem instances. For each instance, its identifier is shown (comprising its number of vertices, tree outspread bound and bound for vertex weights). Additionally, there is the (truly minimal) maximal regret obtained by CPLEX and the maximal regrets computed by the simple and the extended algorithm, respectively. The results of the extended algorithm have been obtained with the parameter values $\pi = 0.2$ and $\delta = 1.0$. Note that for 21 out of 30 instances, the simple algorithm reaches the same maximal regret as CPLEX. For 8 out of 9 remaining instances the extended algorithm improves the result obtained by the simple algorithm, thus reaching again the minimal maximal regret found by CPLEX. There is only one instance where both our approximation algorithms fail to produce the optimal solution. For each approximate solution the table also shows its empirical approximation ratio, i.e., the quotient of its maximal regret versus the minimal maximal regret. The average approximation ratio is 1.12 for the simple algorithm, and 1.01 for the extended algorithm. This is much better than the worst-case approximation ratio 2 from Proposition 4.

Next, we will explain in more detail Table 2. Each row of that table corresponds to one subgroup of 30 medium-sized problem instances with a particular tree outspread bound. All instances within the same subgroup have identifiers with the same prefix, as shown in the first column of the table. The prefix comprises the number of vertices (i.e., 500), the tree outspread bound (i.e., at most 3, 5, 10 and 15 children per vertex, respectively), and the bound for vertex weights (between 1 and at most 50).

Table 2 does not contain truly optimal solutions since the instances are too large to be solved exactly by CPLEX. Instead, only the average maximal regrets obtained by the two approximation algorithms are shown, together with their corresponding average computing times in milliseconds. Additionally, there are the average improvements achieved by the extended algorithm versus the simple algorithm, and the average slowdowns of the extended algorithm versus the simple algorithm. Thereby the improvement for a particular problem instance is computed as the difference of the two corresponding maximal regrets divided by the larger maximal regret. Similarly, the slowdown for a particular problem instance is obtained as the larger computing time divided by the smaller computing time. All values for individual problem instances (maximal regrets, times, improvements, slowdowns) are averaged over each subgroup of 30 problem instances.

Table 2 also presents the concrete values of parameters π and δ used within the extended algorithm. Thereby, for each subgroup of problem instances, different values have been chosen. In fact, in our experiments, we have tested various combinations of π and δ:

$$\begin{aligned}\pi &= \{0.1, 0.2, 0.3, ..., 1\} \cup \{0.01, 0.03, ..., 0.09\} \cup \{0.002, 0.005\} \\ \delta &= \{0.1, 0.2, 0.3, ..., 1\}\end{aligned}$$

The values shown in the table are those that assured the largest average improvements for particular subgroups. One can see that the extended algorithm always improves the results of the simple algorithm, but this improvement is paid by much longer computing time.

From Table 2, one can also observe that improvement usually becomes better (larger) when tree outspread becomes smaller. The best improvement is achieved on trees whose vertices have up to 5 children, while the worst improvement occurs when vertices can have 15 children. On the other hand, slowdowns behave in the opposite way, i.e., they are better (smaller) for trees with larger outspread. Indeed, the best slowdown is accomplished if vertices can have 15 children, and the worst slowdown is for trees whose vertices do not have more than three children. This behavior can be explained as follows: when the total number of vertices is fixed, trees with larger outspread have less levels. And less levels means less independent sets, so there are less combinations to try. The extended algorithm will finish faster because the generated neighborhoods will contain small numbers of new independent sets.

Finally, let us say something about Table 3. It deals with large problem instances, but otherwise is analogous to Table 2. Again, one can see the average improvements and slowdowns of the extended algorithm versus the simple algorithm depending on the tree outspread bound. There is a similar correlation of improvement or slowdown versus tree outspread as in Table 2. Compared to Table 2, average improvements are roughly the same, but they are achieved with much smaller values for π. In fact, average improvements for medium-sized problem instances are optimal when $\pi \in [0.3, 1]$, while for large instances they are optimal when $\pi \in [0.005, 0.1]$. These two intervals do not have intersection. The perceived phenomenon regarding values of π can be explained in the following way. In order to get a new independent set, we need to change the current independent set, but the change must not be too drastic since the current independent set is already a good solution. Large problem instances have much more vertices, thus the same total amount of change is obtained with a smaller value of π. If one compares values of δ from Tables 2 and 3, one can see that there is no such big difference as for values of π. Thus, for both medium-sized and large problem instances, changes of vertex weights (when they occur) should have similar intensity.

Table 1. Results of experiments—small problem instances.

Instance Identifier	CPLEX-Max Regret	Simple Algorithm—Max Regret	Simple Algorithm—Approx Ratio	Extended Algorithm—Max Regret	Extended Algorithm-Approx Ratio
20_3_5_1	6	6	1	6	1
20_3_5_2	5	7	1.4	5	1
20_3_5_3	7	10	1.43	7	1
20_3_5_4	4	5	1.25	4	1
20_3_5_5	3	3	1	3	1
20_3_5_6	4	5	1.25	5	1.25
20_3_5_7	7	10	1.43	7	1
20_3_5_8	5	7	1.4	5	1
20_3_5_9	3	3	1	3	1
20_3_5_10	4	5	1.25	4	1
20_3_5_11	2	2	1	2	1
20_3_5_12	1	1	1	1	1
20_3_5_13	6	6	1	6	1
20_3_5_14	2	2	1	2	1
20_3_5_15	1	1	1	1	1
20_3_5_16	4	4	1	4	1
20_3_5_17	2	2	1	2	1
20_3_5_18	4	5	1.25	4	1
20_3_5_19	4	4	1	4	1
20_3_5_20	1	1	1	1	1
20_3_5_21	2	2	1	2	1
20_3_5_22	3	3	1	3	1
20_3_5_23	3	3	1	3	1
20_3_5_24	5	5	1	5	1
20_3_5_25	2	2	1	2	1
20_3_5_26	1	2	2	1	1
20_3_5_27	3	3	1	3	1
20_3_5_28	4	4	1	4	1
20_3_5_29	6	6	1	6	1
20_3_5_30	2	2	1	2	1
Average			1.12		1.01

Table 2. Results of experiments—medium-sized problem instances.

Instance Identifiers	Perturbation Probability π	Perturbation Intensity δ	Simple Algorithm: Avg Max Regret, Avg Time (ms)	Extended Algorithm: Avg Max Regret, Avg Time (ms)	Avg Improvement (%)	Avg Slowdown (×)
500_3_50_1, 500_3_50_2, ..., 500_3_50_30	0.3	0.8	563.83, 0.572	523.70, 204.562	6.92	368.65
500_5_50_1, 500_5_50_2, ..., 500_5_50_30	0.8	0.5	260.13, 0.534	238.20, 122.849	7.55	233.67
500_10_50_1, 500_10_50_2, ..., 500_10_50_30	1.0	0.7	97.43, 0.508	93.83, 77.400	3.35	154.45
500_15_50_1, 500_15_50_2, ..., 500_15_50_30	0.6	0.6	53.70, 0.466	50.77, 65.261	3.33	140.86

Table 3. Results of experiments—large problem instances.

Instance Identifiers	Perturbation Probability π	Perturbation Intensity δ	Simple Algorithm: Avg Max Regret, Avg Time (ms)	Extended Algorithm: Avg Max Regret, Avg Time (ms)	Avg Improvement (%)	Avg Slowdown ($\times$)
10000_3_1000_1, 10000_3_1000_2, ... , 10000_3_1000_30	0.005	1	222304.23, 42.877	206120.87, 197213.616	7.27	4606.52
10000_5_1000_1, 10000_5_1000_2, ... , 10000_5_1000_30	0.01	1	103190.80, 48.395	97932.13, 99002.375	5.03	2048.50
10000_10_1000_1, 10000_10_1000_2, ... , 10000_10_1000_30	0.1	0.6	34275.20, 45.168	33040.10, 25593.548	3.52	566.14
10000_15_1000_1, 10000_15_1000_2, ... , 10000_15_1000_30	0.1	0.9	19133.20, 44.829	18680.03, 17154.839	2.41	382.99

It is interesting to note that average improvements in Tables 2 and 3 stay similar, although large problem instances contain 20 times more vertices than medium-sized instances. In more detail, average improvements for medium-sized instances are slightly better on subgroups with five and 15 children, while for large instances they are slightly better on subgroups with three and 10 children. On the other hand, average slowdown does not increase 20 times on large instances. In fact, it increases between 2.4 and 12.5 times. The greatest increase is for trees with small outspread and it becomes smaller as the outspread grows. The observed nonlinearity assures that the extended algorithm can be used for very large trees, even larger than those involved in our experiments.

7. Conclusions

In this paper, we have studied two robust variants of the MWIS problem, i.e., the max–min and the min–max regret variant. Both of them have been posed on trees. Uncertainty has been restricted to vertex weights and represented by intervals. We have been interested in complexity aspects, as well as in algorithmic aspects.

On one hand, we have observed that the considered max–min variant can easily be solved in linear time, by an algorithm originally intended for the conventional (non-robust) variant. On the other hand, we have proved that the corresponding min–max regret variant is NP-hard, which justifies its approximate solution. Consequently, the algorithmic part of the paper has been devoted to two approximation algorithms for the min–max regret variant. They are called simple and extended algorithm, respectively. The first of them gives the solution based on the so-called average scenario, while the second one tries to improve that solution by local search.

The simple algorithm runs on our problem instances in linear time. However, its general worst-case approximation ratio 2 cannot be improved in spite of restriction to trees. Still, the experiments presented in the paper clearly indicate that the corresponding average approximation ratio, at least on small trees, is much better, i.e., slightly larger than 1. Regarding the extended algorithm, the experiments have confirmed that it produces even better solutions than the simple algorithm. Indeed, on large trees it improves accuracy of the simple algorithm up to 7%. However, this improvement in accuracy is paid by much larger computing time.

It is important to note that the min–max regret variant of the MWIS problem under interval uncertainty is fairly demanding in computational sense, even when restricted to trees. Namely, the number of implicitly given scenarios that must be considered grows exponentially with the number of vertices. According to our experience, state-of-the-art computers with available software are able to find exact solutions only on rather small trees having few dozens of vertices. At this moment, our approximation algorithms seem to be the only known solution methods that are suitable for larger trees.

In our future work, we will try to speed up the extended algorithm by parallel processing. It can be done since each iteration of the involved local search consists of parallelizable computations. Our second plan for the future is to develop an iterative exact algorithm for the min–max regret variant. The idea is to use a mathematical model, where an initial (small) constraint set is iteratively extended as long as the corresponding solution is infeasible. We expect that such an algorithm will be able to find exact solutions for problem instances of moderate size.

Author Contributions: Conceptualization, formal analysis, software, A.K.; conceptualization, formal analysis, funding acquisition, supervision, validation, writing—original draft, R.M. All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been fully supported by Croatian Science Foundation under the project IP-2018-01-5591.

Data Availability Statement: The data or code used in this research are available in a repository, as specified in the paper.

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

References

1. Gross, J.L.; Yellen, J.; Zhang, P. *Handbook of Graph Theory*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2014.
2. Jungnickel, D. *Graphs, Networks and Algorithms*, 4th ed.; Springer: Berlin, Germany, 2013.
3. Aissi, H.; Bazgan, C.; Vanderpooten, D. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.* **2009**, *197*, 427–438. [CrossRef]
4. Ben-Tal, A.; El Ghaoui, L.; Nemirovski, A. *Robust Optimization*; Princeton University Press: Princeton, NJ, USA, 2009.
5. Bertsimas, D.; Brown, D.B.; Caramani, C. Theory and applications of robust optimization. *SIAM Rev.* **2011**, *53*, 464–501. [CrossRef]
6. Kasperski, A.; Zielinski, P. Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*; Doumpos, M., Zopounidis, C., Grigoroudis, E., Eds.; Springer: Cham, Switzerland, 2016; pp. 113–143.
7. Kouvelis, P.; Yu, G. *Robust Discrete Optimization and Its Applications*; Springer: Berlin, Germany, 1997.
8. Kasperski, A.; Zielinski, P. Complexity of the robust weighted independent set problems on interval graphs. *Optim. Lett.* **2015**, *9*, 427–436. [CrossRef]
9. Klobučar, A.; Manger, R. Independent sets and vertex covers considered within the context of robust optimization. *Math. Commun.* **2020**, *25*, 67–86.
10. Klobučar, A.; Manger, R. An evolutionary algorithm for the robust maximum weighted independent set problem. *Automatika* **2020**, *61*, 523–536. [CrossRef]
11. Klobučar, A.; Manger, R. Solving robust variants of the maximum weighted independent set problem on trees. *Mathematics* **2020**, *8*, 16. [CrossRef]
12. Talla Nobibon, F.; Leus, R. Robust maximum weighted independent-set problems on interval graphs. *Optim. Lett.* **2014**, 8, 227–235. [CrossRef]
13. Korte, B; Vygen, J. *Combinatorial Optimization—Theory and Algorithms*, 5th ed.; Springer: Berlin, Germany, 2012.
14. Saha, A.; Pal, M.; Pal, T.K. Selection of programme slots of television channels for giving advertisement: A graph theoretic approach. *Inf. Sci.* **2007**, 177, 2480–2492. [CrossRef]
15. Barth, L.; Niedermann, B.; Nöllenburg, M. Temporal map labeling: A new unified framework with experiments. In Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in GeographicInformation Systems (SIGSPATIAL 2016), San Francisco, CA, USA, 31 October–3 November 2016; Ali, M., Newsam, S., Ravada, S., Eds.; ACM: New York, NY, USA, 2016; pp. 1–10.
16. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completness*; W.H. Freeman: San Francisco, CA, USA, 1979.
17. Chen, G.H.; Kuo, M.T.; Sheu, J.P. An optimal time algorithm for finding a maximum weight independet set in a tree. *BIT* **1988**, *28*, 253–256. [CrossRef]

18. Pal, M.; Bhattacharjee, G.P. A sequential algorithm for finding a maximum weight k-independent set on interval graphs. *Int. Comput. Math.* **1996**, 60, 205–214. [CrossRef]
19. Williamson, D.P.; Shmoys, D.B. *The Design of Approximation Algorithms*; Cambridge University Press: New York, NY, USA, 2011.
20. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization—Algorithms and Complexity*; Dover Publications: Mineola, NY, USA, 1998.
21. Talbi, E.G. *Metaheuristics—From Design to Implementation*; Wiley: Hoboken, NJ, USA, 2009.
22. IBM ILOG CPLEX Optimization Studio. CPLEX User's Manual, Version 12, Release 8. Available online: https://www.ibm.com/docs/en/icos/12.8.0.0 (accessed on 3 May 2021).