J Glob Optim (2014) 60:265–287
DOI 10.1007/s10898-014-0187-x

An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem

Amadeu Almeida Coco · João Carlos Abreu Júnior · Thiago F. Noronha · Andréa Cynthia Santos

Received: 15 January 2013 / Accepted: 2 April 2014 / Published online: 22 April 2014

Abstract The well-known Shortest Path problem (SP) consists in finding a shortest path from a source to a destination such that the total cost is minimized. The SP models practical and theoretical problems. However, several shortest path applications rely on uncertain data. The Robust Shortest Path problem (RSP) is a generalization of SP. In the former, the cost of each arc is defined by an interval of possible values for the arc cost. The objective is to minimize the maximum relative regret of the path from the source to the destination. This problem is known as the *minmax relative regret* RSP and it is NP-Hard. We propose a mixed integer linear programming formulation for this problem. The CPLEX branch-and-bound algorithm based on this formulation is able to find optimal solutions for all instances with 100 nodes, and has an average gap of 17 % on the instances with up to 1,500 nodes. We also develop heuristics with emphasis on providing efficient and scalable methods for solving large instances for the *minmax relative regret* RSP, based on Pilot method and random-key genetic algorithms. To the best of our knowledge, this is the first work to propose a linear formulation, an exact algorithm and metaheuristics for the *minmax relative regret* RSP.

Keywords Robust shortest path · Uncertain data · Heuristics · Mathematical modeling

A. A. Coco · J. C. A. Júnior · T. F. Noronha (✉)
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, UFMG, Avenida Antônio Carlos 6627, Belo Horizonte, MG CEP 31270-901, Brazil
e-mail: tfn@dcc.ufmg.br

A. A. Coco
e-mail: amadeuac@dcc.ufmg.br

J. C. A. Júnior
e-mail: joao.junior@dcc.ufmg.br

A. C. Santos
ICD-LOSI, Université de Technologie de Troyes, 12, rue Marie Curie, CS 42060, 10004 Troyes Cedex, France
e-mail: andrea.duhamel@utt.fr

1 Introduction

Given a connected digraph $G = (V, A)$ with a set V of nodes and a set A of arcs. Each arc $(i, j) \in A$ is associated to a cost $c_{ij} \in \mathbb{R}$. Moreover, let $n = |V|$ and $m = |A|$ be respectively the total number of nodes and arcs of G. The well-known Shortest Path problem (SP) consists in finding a shortest path from a source $s \in V$ to a destination $t \in V$ such that the total cost is minimized. A solution exists if no negative-weight cycle is reachable from s to t. Polynomial-time algorithms are available to solve SP, such as Dijkstra's [14] and Bellman-Ford's [5]. The SP models practical and theoretical problems [18]. However, several shortest path applications rely on uncertain data.

Some strategies can be applied to solve problems under uncertainty as the stochastic programming [39] and the robust optimization [6,27]. The stochastic programming is mostly applied whenever the probability law associated to the uncertain data is known in advance. A drawback of this approach is that it is sometimes difficult to define the probability distribution associated to the uncertain data, or else errors can happen on the parameters estimation. Moreover, stochastic programming cannot be applied when the optimization implies non-repetitive decisions as for infrastructure design, environmental or nuclear accidents, etc. [1]. We refer to [7,34] for works dedicated to the stochastic shortest path problem, which also extends SP since it minimizes the expected total cost.

Robust optimization is an alternative to stochastic programming where the variability of the data is represented by deterministic values. In this work, we focus on robust optimization models where the uncertain data can be modelled by an interval of possible values. We refer to the book [27] for other robust optimization models. The Robust Shortest Path problem (RSP) is a generalization of SP, where the cost of each arc $(i, j) \in A$ is defined by an interval $[l_{ij}, u_{ij}]$, with $l_{ij}, u_{ij} \in \mathbb{Z}$, where $u_{ij} \geq l_{ij} \geq 0$, for all $(i, j) \in A$ [22]. There are different versions of RSP with interval data in the literature, that differ from each other by the optimization criteria used [1,2,10,23,31–33].

The most studied version of RSP uses the *minmax regret* criterion and is called *minmax regret* RSP. Let $P \subseteq A$ be a path from an origin s to a destination t in G. The *regret* of P in the scenario r (also referred as the *robust deviation* of P in r) is defined as the difference between the cost of P in r and the cost of the shortest path S^r from s to t in r. In other words, the *robust deviation* of P in r is the regret of using P instead of S^r in case the scenario r occurs. The *robust cost* of P is the largest robust deviation of P over all scenarios. The *minmax regret* RSP consists in finding the path P^* from s to t with the smallest robust cost. This problem is shown to be NP-hard even for acyclic digraphs [27].

This work is dedicated to the *minmax relative regret* RSP. Let $P \subseteq A$ be a path from an origin s to a destination t in G. The *relative regret* of P in the scenario r (also referred as the *relative robust deviation* of P in r) is defined as $(cost(P, r) - cost(S^r, r))/cost(S^r, r)$, where $cost(P, r)$ denotes the cost of P in r and $cost(S^r, r)$ denotes the cost of S^r in r. The *relative robust cost* of P is the largest relative deviation of P over all scenarios. The *minmax relative regret* RSP consists in finding the path P^* from s to t with the smallest relative robust cost. This problem is shown to be NP-hard even for acyclic digraphs [2]. Although the *minmax relative regret* RSP differs from the *minmax regret* RSP only by the objective function, the former is more difficult to solve, because the objective function is nonlinear.

Relative robust deviation is a better metric than *robust deviation* because the regret of using P instead of S^r is normalized by the cost of S^r. For instance, let two paths P' and P'', as well as two scenarios r' and r'' such that $cost(P', r') = 11$, $cost(S^{r'}, r') = 1$, $cost(P'', r'') = 100$, and $cost(S^{r''}, r'') = 90$. According to the *minmax regret* criterion, the regret of P' in r' $(11 - 1 = 10)$ is the same as that of P'' in r'' $(100 - 90 = 10)$. However, one can see that

the cost of P' is tenfold that of $S^{r'}$, while the cost of P'' is only 11 % larger than that of $S^{r''}$. According to the *minmax relative regret* criterion, the regret of P' in r' $((11-1)/1 = 10)$ is much larger than that of P'' in r'' $((100-90)/90 = 0.11)$.

The remainder of this paper is organized as follows. First, related works are reviewed in Sect. 2. Then, the first Mixed Integer Linear Programming (MILP) formulation and valid inequalities for the *minmax relative regret* RSP is proposed in Sect. 3. Next, polynomial-time heuristics are proposed in Sect. 4. Following, computational results are reported in Sect. 5. Finally, concluding remarks are drawn in the last section.

2 Related works

The survey [1] is dedicated to computational complexity results for several robust optimization problems, while the survey [17] is dedicated to RSP with interval data and RSP with discrete scenarios. A general overview on robust optimization problems and applications, as well as mathematical models for different versions of RSP, is found in [27]. A study on exact algorithms, approximation algorithms and heuristics for several optimization problems, including the RSP, is presented in [10].

The *minmax regret* RSP was shown to be NP-hard in [27]. A MILP formulation for this problem was introduced in [22]. In [33], the authors proposed a branch-and-bound algorithm based on a combinatorial relaxation of the robust constraints. This work was extended in [32] by using a Benders decomposition algorithm based on the same relaxation. Both algorithms solved instances in random graphs with up to 4,000 nodes and real instances with up to 2,500 nodes.

The works [3,16] investigate instances that can be solved in polynomial or pseudo-polynomial time for classic robust optimization problems. A pseudo-polynomial algorithm is proposed in [24] for the *minmax regret* RSP. It works on series-parallel multidigraphs and has computational complexity of $O(m \cdot |S^u|^2)$, where $|S^u|$ stands for the number of arcs in the shortest path from s to t in the scenario where the arc costs are set to u_{ij}. 2-approximation algorithms for a wide class of *minmax regret* optimization problems are presented in [12]. For the *minmax regret* RSP, a 2-approximation algorithm with worst case complexity of $O(m+n\log n)$ is proposed in [24]. An improved version of this algorithm is given in [25] for series-parallel multidigraphs with approximation factor of $(1+\epsilon)$ and worst case complexity of $O(q(n, n/\epsilon))$, where q is a bivariate polynomial function and ϵ is a value in the range [0, 1].

Preprocessing techniques for the *minmax regret* RSP are proposed in [11,22]. The general idea is to remove dominated arcs, called of *weak-arcs*, i.e., those arcs that do not belong to an optimal solution. An algorithm to eliminate *weak-arcs* is introduced in [22] for particular graphs such as acyclic, planar, and layered graphs. In [11], the author extends the preprocessing proposed in [22], by eliminating vertices and by applying other strategies to deduce *weak-arcs*. The procedure to eliminate vertices consumes $O(n^3)$ in the worst case, and it eliminates vertices not appearing in any shortest path between the origin and the destination. Moreover, the authors present strategies using the minimum spanning tree to detect *weak-arcs*, and they have computational complexity of $O(m \cdot n^2)$ in the worst case.

The *minmax relative regret* RSP was introduced in [27]. In [2], the author proved that this version of RSP is NP-hard on acyclic digraphs, and proposed an integer non-linear formulation for the *minmax relative regret* RSP. Finally, as far as we know there is no exact algorithms or linear integer formulation for this version of RSP in the literature.

The contributions of this work to the literature are the following. We propose the first integer linear programming formulation for *minmax relative regret*, as well as valid inequal-

ities. We also added cycle elimination constraints to this formulation and perform computational experiments on the classic acyclic Karasan instances and on a new set of instances based on digraphs with cycles. Moreover, we develop heuristics with emphasis on providing efficient and scalable methods for solving the *minmax relative regret* RSP, based on Pilot method [15,43] and Random-key Genetic Algorithms [4,21]. The ideas and principles applied in these heuristics can be generalized to build new heuristics for other robust optimization problems.

3 Mixed integer linear programming formulation for the *minmax relative regret* RSP

A MILP formulation for the *minmax regret* RSP was proposed in [22]. The authors proved that the worst scenario for a path $\mathcal{P}$ can be obtained by setting costs $c^r_{ij} = u_{ij}$ for all $(i, j) \in \mathcal{P}$, and $c^r_{ij} = l_{ij}$ for all $(i, j) \in A \setminus \mathcal{P}$. Using this result, the MILP formulation was defined with decision variables $y_{ij} = 1$ if arc $(i, j) \in A$ belongs to the solution, and $y_{ij} = 0$ otherwise. Moreover, auxiliary variables $x_i \geq 0$, for all $i \in V$, keep the cost of the shortest path from the source s to node i in the worst case scenario of the path induced by variables y_{ij}. The corresponding MILP formulation is defined by Eqs. (1)–(6).

$$\min \quad z = \sum_{(i,j)\in A} u_{ij} \cdot y_{ij} - x_t \quad \text{subject to:} \tag{1}$$

$$\sum_{(j,k)\in A} y_{jk} - \sum_{(i,j)\in A} y_{ij} = \begin{cases} 1, & \text{if } j = s \\ -1, & \text{if } j = t \\ 0, & \text{otherwise} \end{cases} \quad \forall j \in V \tag{2}$$

$$x_j \leq x_i + l_{ij} + (u_{ij} - l_{ij}) y_{ij} \quad \forall (i, j) \in A \tag{3}$$

$$x_s = 0 \tag{4}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{5}$$

$$x_i \geq 0 \quad \forall i \in V \tag{6}$$

The objective function (1) minimizes the regret at the destination node t. Constraints (2) are the classic flow conservation constraints and ensure the path connectivity from nodes s to t. Inequalities (3) link variables y and x and determine an upper bound on the regret at node j. These constraints together with the objective function ensure the maximum regret to be minimized. Equation (4) sets x_s to zero. The domain of the variables y and x are defined in (5) and (6), respectively. This formulation can be efficiently solved by commercial solvers as CPLEX for some benchmarks. For example, random instances with up to 900 vertices and a real instance, from the road network of Stuttgart, with 2,490 vertices are solved in a few seconds in [33]. Instances with up to 1,000 vertices are solved to optimality, and CPLEX solver runs out of memory for those varying from 10,000 to 20,000 vertices, with gaps within 2–9 % [36].

A mixed integer non-linear formulation for the *minmax relative regret* RSP is proposed in [2]. The constraints are those from (2) to (6), presented above for the *minmax regret* RSP, and the objective function is that of (7).

$$\min \frac{\sum_{(i,j)\in A} u_{ij} y_{ij} - x_t}{x_t} \tag{7}$$

We observe that precautionary measures need to be taken when applying the model proposed in [2] to graphs with cycles. Due to the objective function (7), it might be possible to reduce the regret of a loopless path by adding subcycles to this path. Fig-

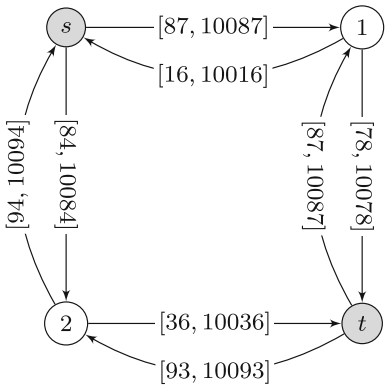

Fig. 1 Example of graph with cost defined in an interval with nonnegative values

ure 1 illustrates such situation, where s and t are respectively the origin and the destination. If subcycles are forbidden, then for this example, two paths exists from s to t: $P^1 = \{(s,1),(1,t)\}$ and $P^2 = \{(s,2),(2,t)\}$, respectively with robust costs $\left(\frac{20165-120}{120} = 167.04\right)$ and $\left(\frac{20120-165}{165} = 120.94\right)$. The formulation (2)–(7) allows a path $P' = \{(s,1),(1,s),\ (s,2),(2,t),(t,1),(1,t)\}$, which is larger than P^1 and P^2 together, but has a smaller robust cost equal to $\left(\frac{60388-20120}{20120} = 2.0014\right)$. The regret of P' is artificially smaller than those of P^1 and P^2, because the arcs in all paths from s to t are in P'. Thus, the corresponding arc costs in the worst case scenario r' of P' are in their corresponding upper bound. This increases the cost of the shortest path $S^{r'}$ and artificially decreases the relative regret of P'. Therefore, we only consider loopless paths in the rest of this paper and enforce the subcycle elimination constraints.

Section 3.1 presents a linearization to (7), while the subcycles elimination constraints are given in Sect. 3.2. Valid inequalities that strengthens the linear relaxation of the resulting formulation are shown in Sect. 3.3.

3.1 Linear formulation

The objective function (7) has been linearized inspired by work [8], for the case where $l_{ij}, u_{ij} \in \mathbb{N}$. First, without loss of generality, we rewrite the objective function as

$$\min \frac{\sum_{(i,j)\in A} u_{ij} y_{ij}}{x_t} - \frac{x_t}{x_t} = \sum_{(i,j)\in A} u_{ij} \frac{y_{ij}}{x_t} - 1 \tag{8}$$

Next, we introduce variables $z_{ij} \in \mathbb{R}$, together with constraints (9)–(11), to ensure that $z_{ij} = \frac{y_{ij}}{x_t}$, for all $(i,j) \in A$.

$$z_{ij} \leq y_{ij} \quad \forall (i,j) \in A \tag{9}$$

$$z_{ij} \geq \frac{1}{x_t} - (1 - y_{ij}) \quad \forall (i,j) \in A \tag{10}$$

$$z_{ij} \in \mathbb{R} \quad \forall (i,j) \in A, \tag{11}$$

Note that constraints (10) remain nonlinear. They can be linearized with the introduction of variables w_l, for all $l \in \{L,\ldots,U\}$, such that $w_l = 1$ if and only if $x_t = l$, and $w_l = 0$ otherwise. The lower bound L to the value of x_t is defined as the cost of the shortest path

from s to t in the scenario where the arc costs are set to $c^l_{ij} = l_{ij}$, and the upper bound U to the value of x_t is defined as the cost of the shortest path from s to t in the scenario where the arc costs are set to $c^u_{ij} = u_{ij}$. Therefore, the number of w_l variable is equal to $U - L$. Then, we replace constraints (10) by constraints (12)–(15).

$$z_{ij} \geq \sum_{l \in \{L,\ldots,U\}} \frac{1}{l} \cdot w_l - (1 - y_{ij}) \quad \forall (i,j) \in A \tag{12}$$

$$\sum_{l \in \{L,\ldots,U\}} w_l \geq 1 \tag{13}$$

$$\sum_{l \in \{L,\ldots,U\}} l \cdot w_l = x_t \tag{14}$$

$$w_l \in \{0,1\} \quad \forall l \in \{L,\ldots,U\} \tag{15}$$

Finally, we can rewrite the nonlinear objective function (7) as the linear objective function (16), and a MILP formulation for the *minmax relative regret* RSP can be defined by Eqs. (2)–(6), (9), (11)–(16). The number of constraints in this formulation is $O(|A|)$, while the number of variables is $O(|A| + U)$.

$$\min \sum_{(i,j) \in A} u_{ij} z_{ij} - 1 \tag{16}$$

3.2 Subcycle elimination constraints

The subcycle elimination constraints proposed in [30] have been successfully applied for a number of problems in the literature [13,38], and we have adapted them for the *minmax relative regret* RSP. In the context of the *minmax relative regret* RSP, such constraints establish a topological order for each visited node in the robust path from s to t, and thus eliminate subtours.

The constraints (17)–(19) make use of variables t_i, $\forall i \in V$, which determine the order vertex i appears in the robust path. Restrictions (17) ensure that if arc $(i, j) \in A$ is chosen in the path, i.e. if $y_{ij} = 1$, then $t_i < t_j$. M is set to $|V|$ since a path from s to t without cycles has up to $|V|$ nodes. Equality (18) sets that the origin s has $t_s = 0$. Finally, the domain of variables t_i are defined in (19), $\forall i \in V \backslash \{s\}$. For example, given a path $\mathcal{P} = \{(s, 1), (1, s), (s, 2), (2, t)\}$, whenever arcs $(s, 1)$ and $(1, s)$ are simultaneously considered, the corresponding constraints (17) assume respectively $t_s - t_1 \leq -1$ and $t_1 - t_s \leq -1$. But, these inequalities cannot be simultaneously satisfied. Thus, in such a way, subcycles are eliminated.

The complete formulation considered here for the *minmax relative regret* is given by the objective function (16) and constraints (2)–(6), (9), (11)–(19).

$$t_i - t_j + M y_{ij} \leq M - 1, \qquad \forall (i,j) \in A \tag{17}$$

$$t_s = 0 \tag{18}$$

$$0 \leq t_i \leq M \qquad \forall i \in V \setminus \{s\} \tag{19}$$

3.3 Valid inequalities

Given the lower bound L and the upper bound U for the value of x_t defined above, it follows that

$$\frac{\sum_{(i,j) \in A} u_{ij} y_{ij} - x_t}{U} \leq \frac{\sum_{(i,j) \in A} u_{ij} y_{ij} - x_t}{x_t} \leq \frac{\sum_{(i,j) \in A} u_{ij} y_{ij} - x_t}{L}.$$

We have that

$$\sum_{(i,j)\in A} u_{ij} z_{ij} - 1 = \frac{\sum_{(i,j)\in A} u_{ij} y_{ij} - x_t}{x_t}$$

in an optimal integer solution for the MILP formulation described above. Therefore, the inequalities (20)–(21) are valid for this formulation. Although (20)–(21) are redundant in the integer formulation, they are a stronger link between variables z_{ij}, y_{ij}, and x_t, which improves (i) the lower bound of the corresponding linear relaxation and (ii) the performance of the corresponding CPLEX branch-and-bound algorithm.

$$\sum_{(i,j)\in A} u_{ij} z_{ij} - 1 \geq \frac{\sum_{(i,j)\in A} u_{ij} y_{ij} - x_t}{U} \tag{20}$$

$$\sum_{(i,j)\in A} u_{ij} z_{ij} - 1 \leq \frac{\sum_{(i,j)\in A} u_{ij} y_{ij} - x_t}{L} \tag{21}$$

The resulting formulation for the *minmax relative regret* is given by the objective function (16) and constraints (2)-(6), (9), (11)-(21). These constraints not only improve the linear relaxation by imposing a lower bound to the value of the objective function, but also by strengthening the link between variables z_{ij}, y_{ij}, and x_t.

4 Heuristics for RSP

In [24,25], three approximated solutions were proposed for the *minmax regret* RSP. The first, called AM, fixes the cost of all arcs to their respective mean value (i.e. $(l_{ij} + u_{ij})/2$) and returns the shortest path in this scenario (called the *Median Scenario*). Algorithm AM is a 2-approximation algorithm [25]. The second, called AU, fixes the cost of all arcs to their corresponding maximum value (i.e. u_{ij}) and returns the shortest path in this scenario (called the *Upper Scenario*). The third, called AMU, returns the best solution found by AM and AU. Therefore, AMU is also 2-approximative [25]. No experimental results are presented to address the performance of these heuristics on the average case.

A Simulated Annealing heuristic inspired by [26] was proposed in [36] for the *minmax regret* RSP. It is called here SA-RSP. The parameters of SA-RSP are (i) the initial temperature $t_0 \in \mathbb{R}$, (ii) the final temperature $t_f \in \mathbb{R}$, (iii) the temperature decreasing rate β (with $0 < \beta < 1$), and (iv) a value $\lambda \in \mathbb{N}$ that sets the number of accepted solution before decreasing the temperature. Initially, SA-RSP runs the AU heuristic to obtain an initial feasible solution. Then, the procedure performs the four following steps at each iteration. First, it selects a subset $A' \subset A$ at random. Next, it runs the AU heuristic in the graph $G' = (V, A')$ in order to obtain a candidate solution P'. Then, P' is accepted if $cost(\mathcal{P}') < cost(\mathcal{P})$ or else if $e^{\frac{-\Delta}{t}} < \theta$, where $cost(\mathcal{P})$ and $cost(\mathcal{P}')$ are the cost of paths $\mathcal{P}$ and $\mathcal{P}'$, respectively, $\Delta = cost(\mathcal{P}') - cost(\mathcal{P})$, $\theta \in [0, 1[$ is a randomly generated number, and t is the current temperature of the simulated annealing. Following, $t = t * \beta$, if λ solutions had been accepted since the last update took place. Finally, the procedure stops if $t \leq t_f$, and the best solution found so far is returned.

We propose below new heuristics for the *minmax relative regret* RSP. First, a constructive heuristic based on the Pilot Method metaheuristic is described in Sect. 4.1. Then, a genetic algorithm [4] is proposed in Sect. 4.2. These heuristics are compared with AMU [25] and SA-RSP [36] in Sect. 5.

4.1 Pilot method for RSP

Pilot method [15,43] is a metaheuristic that uses a greedy constructive (guiding) heuristic H to build a new and more efficient heuristic H'. The latter works as a traditional constructive heuristic that iteratively inserts one element at a time in a partial solution. However, instead of using a local greedy criterion to evaluate the cost of inserting an element in the solution, the criterion used by H' consists in (i) inserting the element individually in the solution (ii) executing the heuristic H until a feasible solution is found, and (iii) using the cost of this solution as the greedy cost of inserting the element. At each iteration, these three steps are executed for all candidate elements and the one with the best greedy cost is inserted on the solution.

The Pilot method heuristic proposed for the *minmax relative regret* RSP is called PM-RSP. The guiding heuristic used is based on the AM heuristic [24,25]. The pseudo-code of PM-RSP is presented in Algorithm 1. It takes as input a graph $G = (V, A)$, the lower bound l_{ij}, the upper bound u_{ij}, to the cost of each arc $(i, j) \in A$, the origin $s \in V$ and the destination $t \in V$. The partial solution P' (also called guiding solution) and best known feasible solution $\mathcal{P}^*$ are initialized in line 1. The loop in lines 2-15 is performed while $\mathcal{P}'$ is not a path from s to t, i.e., while node t is not inserted in $\mathcal{P}'$. The node u that was last inserted in $\mathcal{P}'$ is identified in line 3. The loop in lines 4-10 is performed for each node $v \in \delta^+(u)$, where $\delta^+(u)$ is set of the successors of vertex u. In line 5, the Dijkstra's algorithm is applied to get the shortest path $\mathcal{P}_v$ from v to t in the scenario r^m. Next, paths $\mathcal{P}'$ (between s and u) and $\mathcal{P}_v$ (between v and t) are concatenated in line 6, forming the path $\mathcal{P}'_v$ (from s to t). The relative robust cost of $\mathcal{P}'_v$ is used as the greedy cost of inserting node v in the solution $\mathcal{P}'$. Then, if the relative robust cost of $\mathcal{P}'_v$ is smaller than that of $\mathcal{P}'_{v^*}$ or else if the latter is not set yet (line 7), the node $v^* \in \delta^+(u)$ with the smallest greedy cost and its respective path $\mathcal{P}'_{v^*}$ are updated in line 8. Afterwards, v^* is inserted at the end of $\mathcal{P}'$ in line 11. PM-RSP returns the best solution found throughout the heuristic $\mathcal{P}^*$, which is not necessarily $\mathcal{P}'$. Therefore, the former is updated in lines 12-13, and returned in line 16. The worst case complexity of PM-RSP is equal to

```
   Input: G = (V, A), l_ij, u_ij, s, t
   Output: P*
1  P' ← s and P* ← ∅
2  P'_v ← ∅
3  while P' is not a path between s and t do
4  |  Let u be the last node inserted in P'
5  |  for v ∈ δ+(u) do
6  |  |  P_v ← Dijkstra (v, t, G, r^m)
7  |  |  P'_v ← Concatenate (P', P_v)
8  |  |  if cost (P'_v) < cost(P'_v*) or P'_v* = ∅ then
9  |  |  |  v* ← v and P'_v* ← P'_v
10 |  |  end
11 |  end
12 |  Insert v* at the end of P'
13 |  if cost(P'_v*) < cost(P*) then
14 |  |  P* ← P'_v*
15 |  end
16 end
17 return P*;
```

Algorithm 1: Pseudo-code for PM-RSP

$|A|$ times the complexity of the Dijkstra's algorithm, that is $(\mathcal{O}(|A| \cdot (|A| + |V| \cdot \log |V|)))$. Assuming that $|A| \leq |V| \cdot \log |V|$, we have $\mathcal{O}(|A| \cdot |V| \cdot \log |V|)$.

The choice of the Pilot method metaheuristic to develop a heuristic for the *minmax relative regret* RSP is motivated by its successful applications to many combinatorial optimization problems (see [43] for a survey). Besides, Pilot method is a natural framework for developing heuristics for robust optimization problems with interval data, because any efficient and well known algorithm for the static optimization counterpart can be applied to the median or upper scenario and used as the guiding heuristic for a PM heuristic to the robust counterpart.

4.2 Biased random-key genetic algorithm for RSP

Genetic algorithms with random keys, or random-key genetic algorithms (RKGA), were first introduced by Bean [4] for combinatorial optimization problems for which solutions can be represented as a permutation vector. Solutions are represented as vectors of randomly generated real numbers called keys. A deterministic algorithm, called a decoder, takes as input a solution vector and associates with it a feasible solution of the combinatorial optimization problem, for which an objective value or fitness can be computed. Two parents are selected at random from the entire population to implement the crossover operation in the implementation of a RKGA. Parents are allowed to be selected for mating more than once in a given generation.

A biased random-key genetic algorithm (BRKGA) differs from a RKGA in the way parents are selected for crossover. In a BRKGA, each element is generated combining one element selected at random from the elite solutions in the current population, while the other is a non-elite solution. We say the selection is biased since one parent is always an elite individual and because this elite solution has a higher probability of passing its genes to the offsprings, i.e. to the new generation. A BRKGA provides a better implementation of the essence of Darwin's principle of "survival of the fittest", since an elite solution has a higher probability of being selected for mating and the offsprings have a higher probability of inheriting the genes of the elite parent.

The BRKGA proposed for the *minmax relative regret* RSP (BRKGA-RSP) evolves a population of chromosomes that consists of vectors of real numbers in the range $[0, 1]$ (called keys), which are randomly generated in the initial population. The fitness of the chromosome is given by the cost of the solution found by a decoding heuristic that receives as input the random-keys vector and outputs a feasible solution with its corresponding cost. Each chromosome p has one key $k_{ij}^p \in [0, 1]$ for each arc $(i, j) \in A$, and the decoding heuristic performs the following steps. First, it fixes the cost of all arcs to $c_{ij}^p = l_{ij} + (u_{ij} - l_{ij}) \cdot k_{ij}^p$, for all $(i, j) \in A$, i.e. to the scenario induced by the keys of p. This scenario is called r^p. Next, Dijkstra's algorithm [14] is used to compute the shortest path $\mathcal{P}^p$ between s and t in the scenario r^p. Then, the relative robust cost of $\mathcal{P}^p$ is used as the fitness of the chromosome.

The *parameterized uniform crossover* scheme proposed in [40] is used to combine two parent solutions and produce an offspring solution. In this scheme, the offspring inherits each of its keys from the respective key from one of its two parents with probability 0.5.

This genetic algorithm does not make use of the standard mutation operator, where parts of the chromosomes are changed with small probability. Instead, the concept of *mutants* is used. In each generation, a fixed number of mutant solutions are introduced in the population. They are generated in the same way as the initial population. As with mutation, mutants serve the role of helping the procedure escape from local optima.

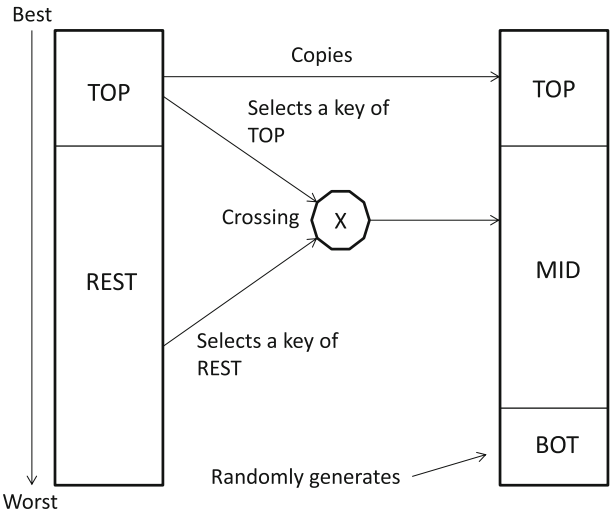

Fig. 2 Illustration of the transitional process between consecutive generations of the biased random-key genetic algorithms

At each new generation, the population is partitioned into two sets: *TOP* and *REST*. Consequently, the size of the population is $|TOP| + |REST|$. The best solutions are kept in *TOP* while the others are placed in *REST*. As illustrated in Fig. 2, the chromosomes in *TOP* are copied, without change, to the population of the next generation. The new mutants are placed in a set called *BOT*. The remaining elements of the new population are obtained by crossover with one parent randomly chosen from *TOP* and the other from *REST*. This distinguishes a biased random-key GA of [19] from the random-key GA of Bean [4]. In the latter both parents are selected at random from the entire population. Since a parent solution can be chosen for crossover more than once in a given generation, elite solutions have a higher probability of passing their random-keys to the next generation. In this way, $|REST| - |BOT|$ offspring solutions are created. The algorithm stops when a maximum number of generations is reached. The relative sizes of sets *TOP*, *REST*, and *BOT* are parameters that must be tuned. However, it was observed in [20,29,35,37] that the performance of this metaheuristic is not significantly sensitive to the value of these parameters.

The choice of the BRKGA metaheuristic to develop a heuristic for the *minmax relative regret* RSP is motivated by its successful applications to many combinatorial optimization problems (see [19] for a survey). Besides, BRKGA is also a natural framework for developing heuristics for robust optimization problems with interval data, because the chromosomes represent scenarios, instead of solutions. The specific solutions for a particular robust optimization problem are obtained by the decoding heuristic that consists of any efficient and well known algorithm for the static optimization counterpart, applied to the scenarios represented by the chromosomes.

5 Computational experiments

Computational experiments were performed on an Intel Core i7 machine with 2.67 GHz clock and 4 GB of RAM memory, running Linux operating system. Two branch-and-bound algorithms for the *minmax relative regret* RSP were implemented making use of ILOG

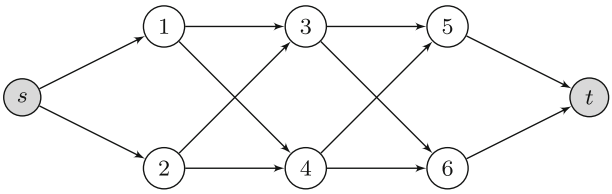

Fig. 3 A Karasan graph with 8 nodes and 3 layers

CPLEX version 12.5 with default parameter settings. The first one based on (2)-(6), (9), (11)-(19) is named CPLEX (V1), and the second one based on (2)-(6), (9), (11)-(21) is named CPLEX (V2) and differs from the first by the valid inequalities (20) and (21). In both cases, constraint (17) was lifted according to [13]. Heuristics AMU [25], SA-RSP [36], PM-RSP, and BRKGA-RSP were implemented in C++ and compiled with GNU GCC version 4.6.3. The pseudorandom number generator used was the Mersenne Twister [28]. Besides, BRKGA-RSP [42] was implemented on the C++ application programming interface for biased random-key algorithms of [42]. Two sets of instances are used in the computational experiments: Karasan instances [22] and the grid instances proposed in this paper. Both sets are described below.

Karasan graphs were proposed in [22] and used in the computational experiments of [22,31–33,36]. They are layered [41] and acyclic [9] graphs that are claimed to resemble telecommunication networks. In a Karasan graph, all the C layers have the same number W of nodes. There is an arc from every node in a layer c to every node in the layer $c+1$, with $c \in \{1, \ldots, C-1\}$. Besides, there is an arc from s to every node in layer 1, and an arc from every node in layer C to t. A Karasan graph with 8 nodes (including s and t) and 3 layers is displayed in Fig. 3. The interval data $[l_{ij}, u_{ij}]$, for all $(i, j) \in A$, are set as following. First, a random number $\theta_{ij} \in [1, \theta_{max}]$ is generated for all $(i, j) \in A$, where $\theta_{max} = 200$. Then, l_{ij} is set to $U[(1-d)\cdot\theta_{ij}, (1+d)\cdot\theta_{ij}]$, and u_{ij} is set to $U[l_{ij}, (1+d)\cdot\theta_{ij}]$, where $d = 0.9$ and $U[a, b]$ denotes a random number uniformly selected in the interval $[a, b]$. The instances are named as K-v-θ_{max}-d-i-W, where K identifies the instance type, v is the number of nodes between s and t, and i distinguishes different instances generated with the same parameter values. The value of the above mentioned parameters θ_{max}, d, and W are also displayed in the instance name.

Grid graphs are based on a $n \times m$ matrix, where n is the number of rows and m is the number of columns. Each matrix cell corresponds to a node and there are two bidirectional arcs between each pair of nodes whose respective matrix cell are adjacent. The source node s is defined as the upper left node and the target node t is defined as the lower right node. These instances were introduced because they resemble street intersection in road networks, and because they contain many cycles. An example of a 3×5 grid is shown in Fig. 4. The value of $[l_{ij}, u_{ij}]$, for all $(i, j) \in A$, was generated the same way as for the Karasan instances. The instances are named as G-$n \times m$-i, where G identifies the instance type, n is the number of rows, m is the number of columns, and i distinguishes different instances generated with the same parameter values.

In the first experiment, we evaluate the impact of the valid inequalities (20) and (21) by comparing the performance of CPLEX (V1) and CPLEX (V2) for small Karasan and grid instances with 100 and 200 nodes. The maximum running time for both algorithms was set to 7,200 s. The results of this experiment are displayed in Tables 1 and 2, respectively. The instance name is displayed in column 1. The lower bound (lb) and the upper bound (ub) to the value of the optimal solution obtained by CPLEX (V1) are given in

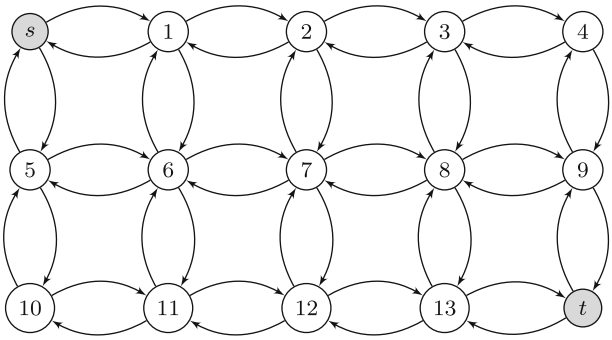

Fig. 4 An example of 3×5 grid

columns 2 and 3, respectively. The integrality gap, the running time, and the number of nodes evaluated in the branch-and-bound tree are reported respectively in columns 4 to 6. The same data are displayed for CPLEX (V2) in the last five columns, respectively. It can be seen that the valid inequalities (20) and (21) improved the linear relaxation lower bounds. Regarding the Karasan instances (Table 1), one can see that the smaller is the number of nodes per layer, the harder was the instance to solve. CPLEX (V1) found optimal solutions for all instances but K-200-200-0.9-a-2, K-200-200-0.9-b-2, K-200-200-0.9-a-5, and K-200-200-0.9-b-5 before 7,200 s, while CPLEX (V2) managed to solve the latter in addition to all the other instances solved by CPLEX (V1). The average relative integrality gap of CPLEX (V1) was 20.19 %, while that of CPLEX (V2) was only 2.40 %. Regarding the grid instances (Table 2), CPLEX (V1) solved all instances with 100 nodes but none with 200 nodes before 7,200 s, while CPLEX (V2) solved all instances with 100 nodes and half of the instances with 200 nodes. Besides, CPLEX (V1) failed to find feasible solutions for four (out of the 10) instances with 200 nodes before 7,200 s, while CPLEX (V2) found feasible solutions for all instances. The average relative integrality gap of CPLEX (V1) for these instances was 47.43 %, while that of CPLEX (V2) was only 2.52 %. These results suggest that the grid instances proposed in this paper are harder to solve than the Karasan instances. Besides, they indicate that CPLEX (V2) can efficiently be applied to small Karasan and grid instances with up to 200 nodes.

In the second experiment, we compare CPLEX (V1) and CPLEX (V2) for large Karasan and grid instances with up to 1,500 nodes. The maximum running time for both algorithms was set again to 7,200 s. The results of this experiment are displayed in Tables 3 and 4, respectively. The instance name is displayed in column 1. The lower bound (lb) and the upper bound (ub) to the value of the optimal solution obtained by CPLEX (V1) are given in columns 2 and 3, respectively. The integrality gap, the running time, and the number of nodes evaluated in the branch-and-bound tree are reported respectively in columns 4 to 6. The same data are displayed for CPLEX (V2) in the last five columns, respectively. It can be seen that the valid inequalities (20) and (21) greatly improved the linear relaxation lower bounds. Regarding the Karasan instances (Table 3), CPLEX (V1) solved only one instance, while CPLEX (V2) managed to solve six (out of the 20) instances before 7,200 s. The average relative integrality gap of CPLEX (V1) was 1,051.98 %, while that of CPLEX (V2) was only 16.68 %. Regarding the grid instances (Table 4), no instance was solved by CPLEX (V1) and CPLEX (V2) before 7,200 s. However, the latter found integer feasible solutions for all 10 instances with an average gap of 18.78 %, while the former found no feasible solution for instances in this set. The fact that CPLEX (V2) found solutions with almost 20 % average gap

Table 1 Comparison between CPLEX (V1) and CPLEX (V2) for Karasan instances with 100 and 200 nodes

Instance	CPLEX (V1)					CPLEX (V2)				
	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb(%)	ub (%)	gap(%)	t(s)	Nodes
K-100-200-0.9-a-2	0.00	0.00	0.00	0.87	1,708	0.00	0.00	0.00	0.19	1
K-100-200-0.9-b-2	0.00	0.00	0.00	1.09	2,313	0.00	0.00	0.00	0.17	1
K-100-200-0.9-a-5	2.27	2.27	0.00	0.85	52	2.27	2.27	0.00	0.29	1
K-100-200-0.9-b-5	0.79	0.79	0.00	0.97	209	0.79	0.79	0.00	0.33	1
K-100-200-0.9-a-10	0.63	0.63	0.00	1.54	64	0.63	0.63	0.00	0.42	1
K-100-200-0.9-b-10	0.00	0.00	0.00	0.56	4	0.00	0.00	0.00	0.19	1
K-100-200-0.9-a-25	0.00	0.00	0.00	0.42	1	0.00	0.00	0.00	0.31	1
K-100-200-0.9-b-25	1.35	1.35	0.00	0.74	1	1.35	1.35	0.00	0.49	1
K-100-200-0.9-a-50	2.44	2.44	0.00	0.44	1	2.44	2.44	0.00	2.04	1
K-100-200-0.9-b-50	0.00	0.00	0.00	0.64	1	0.00	0.00	0.00	0.46	1
K-200-200-0.9-a-2	−86.38	60.99	147.37	7,200.00	1,202,871	39.35	60.93	21.58	7,200.00	1,199,976
K-200-200-0.9-b-2	−80.98	59.81	140.79	7,200.00	521,026	38.65	59.68	21.03	7,200.00	1,480,963
K-200-200-0.9-a-5	−9.77	62.37	72.15	7,200.00	1,175,419	54.90	60.20	5.30	7,200.00	586,265
K-200-200-0.9-b-5	−4.35	39.22	43.57	7,200.00	1,027,303	39.22	39.22	0.00	55.14	19,681
K-200-200-0.9-a-10	73.08	73.08	0.00	2,453.23	445,310	73.08	73.08	0.00	46.75	16,325
K-200-200-0.9-b-10	72.09	72.09	0.00	1,253.04	321,462	72.09	72.09	0.00	9.93	3,077
K-200-200-0.9-a-25	45.83	45.83	0.00	8.79	76	45.83	45.83	0.00	4.47	4
K-200-200-0.9-b-25	76.92	76.92	0.00	11.55	209	76.92	76.92	0.00	5.42	77
K-200-200-0.9-a-50	35.71	35.71	0.00	3.46	1	35.71	35.71	0.00	5.01	1
K-200-200-0.9-b-50	25.00	25.00	0.00	6.28	1	25.00	25.00	0.00	5.56	1
Average			20.19					2.40		

Table 2 Comparison between CPLEX (V1) and CPLEX (V2) for grid instances with 100 and 200 nodes

Instance	CPLEX (V1)					CPLEX (V2)				
	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb (%)	ub (%)	gap (%)	t(s)	Nodes
G_3 × 30_a	0.50	0.50	0.00	1.33	118	0.50	0.50	0.00	0.58	1
G_3 × 30_b	0.70	0.70	0.00	1.91	655	0.70	0.70	0.00	0.70	22
G_4 × 25_a	0.96	0.96	0.00	2.96	2,539	0.95	0.95	0.00	0.81	1
G_4 × 25_b	0.00	0.00	0.00	2.04	386	0.00	0.00	0.00	0.38	1
G_5 × 20_a	0.63	0.63	0.00	3.48	1,335	0.63	0.63	0.00	0.61	1
G_5 × 20_b	0.00	0.00	0.00	1.66	377	0.00	0.00	0.00	0.29	1
G_6 × 17_a	0.00	0.00	0.00	2.32	92	0.00	0.00	0.00	0.31	1
G_6 × 17_b	0.00	0.00	0.00	0.87	306	0.00	0.00	0.00	0.20	1
G_7 × 14_a	0.00	0.00	0.00	1.29	368	0.00	0.00	0.00	0.30	1
G_7 × 14_b	1.49	1.49	0.00	2.03	464	1.49	1.49	0.00	0.63	2
G_4 × 50_a	−98.99	–	–	7,200.00	176,946	33.31	45.19	11.89	7,200.00	333,410
G_4 × 50_b	−87.98	34.36	122.34	7,200.00	79,988	27.98	34.36	6.38	7,200.00	267,725
G_5 × 40_a	−95.05	–	–	7,200.00	163,330	33.54	40.60	7.06	7,200.00	365,738
G_5 × 40_b	−92.57	–	–	7,200.00	153,183	28.03	28.03	0.00	1,029.16	36,778
G_6 × 34_a	−89.60	33.64	123.24	7,200.00	239,289	33.64	33.64	0.00	327.48	23,130
G_6 × 34_b	−90.15	–	–	7,200.00	111,683	39.99	55.58	15.59	7,200.00	497,787
G_7 × 29_a	−89.81	44.99	134.80	7,200.00	155,818	44.98	44.99	0.00	3,994.81	298,610
G_7 × 29_b	−93.82	56.14	149.95	7,200.00	246,237	45.09	54.27	9.49	7,200.00	323,021
G_8 × 25_a	−62.99	26.88	89.86	7,200.00	223,165	26.88	26.88	0.00	99.96	5,566
G_8 × 25_b	−82.46	56.14	138.60	7,200.00	314,228	40.77	40.77	0.00	439.39	34,039
Average			47.43					2.52		

Table 3 Comparison between CPLEX (V1) and CPLEX (V2) for Karasan instances with 1,000 and 1,500 nodes

Instance	CPLEX (V1)					CPLEX (V2)				
	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb (%)	ub (%)	gap (%)	t(s)	Nodes
K-1000-200-0.9-a-5	−95.48	68.86	164.34	7,200.00	20,799	38.70	61.49	22.79	7,200.00	22,155
K-1000-200-0.9-b-5	−100.00	87.93	187.93	7,200.00	14,008	40.87	67.35	26.48	7,200.00	44,525
K-1000-200-0.9-a-10	−100.00	106.97	206.97	7,200.00	17,959	49.26	93.05	43.79	7,200.00	76,124
K-1000-200-0.9-b-10	−94.75	100.76	195.50	7,200.00	41,556	44.56	76.74	32.19	7,200.00	62,477
K-1000-200-0.9-a-25	−98.63	142.36	240.99	7,200.00	36,759	53.57	65.07	11.50	7,200.00	164,165
K-1000-200-0.9-b-25	−94.43	182.19	276.62	7,200.00	19,798	53.71	77.93	24.22	7,200.00	96,155
K-1000-200-0.9-a-50	−73.13	83.72	156.86	7,200.00	21,969	68.09	68.09	0.00	1,100.99	10,967
K-1000-200-0.9-b-50	−65.75	65.91	131.66	7,200.00	24,498	65.91	65.91	0.00	754.09	5,871
K-1000-200-0.9-a-100	84.21	84.21	0.00	2,432.07	1,851	84.21	84.21	0.00	1,521.14	2,261
K-1000-200-0.9-b-100	−8.53	95.00	103.53	7,200.00	13,828	90.00	90.00	0.00	1,716.96	1,707
K-1500-200-0.9-a-5	−99.72	103.72	203.44	7,200.00	2,175	34.16	53.13	18.97	7,200.00	3,441
K-1500-200-0.9-b-5	−98.47	97.96	196.43	7,200.00	3,204	30.93	45.20	14.27	7,200.00	5,230
K-1500-200-0.9-a-10	−99.11	105.50	204.61	7,200.00	3,158	36.58	57.71	21.13	7,200.00	5,818
K-1500-200-0.9-b-10	−99.61	69.56	169.17	7,200.00	9,404	36.58	56.39	19.81	7,200.00	10,923
K-1500-200-0.9-a-25	−93.05	265.97	359.02	7,200.00	8,026	38.65	52.88	14.23	7,200.00	41,947
K-1500-200-0.9-b-25	−100.00	283.33	383.33	7,200.00	8,626	48.47	87.57	39.10	7,200.00	9,560
K-1500-200-0.9-a-50	−77.73	66.67	144.40	7,200.00	4,338	59.26	59.26	0.00	2,197.76	9,618
K-1500-200-0.9-b-50	−100.00	11,911.43	12,011.43	7,200.00	9	52.65	75.51	22.86	7,200.00	37,770
K-1500-200-0.9-a-100	−99.35	1,171.43	1,270.78	7,200.00	1,169	81.25	81.25	0.00	4,934.37	570
K-1500-200-0.9-b-100	−77.06	4,355.56	4,432.62	7,200.00	9	83.60	105.88	22.28	7,200.00	20,884
Average			1,051.98					16.68		

Table 4 Comparison between CPLEX (V1) and CPLEX (V2) for grid instances with up to 1,000 nodes

Instance	CPLEX (V1)					CPLEX (V2)				
	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb (%)	ub (%)	gap (%)	t(s)	Nodes
G_6 × 60_a	−98.67	–	–	7,200.00	44,388	30.86	42.45	11.59	7,200.00	383,891
G_6 × 60_b	−99.08	–	–	7,200.00	35,895	35.75	50.02	14.28	7,200.00	174,914
G_7 × 70_a	−100.00	–	–	7,200.00	9,720	37.62	58.78	21.17	7,200.00	80,621
G_7 × 70_b	−100.00	–	–	7,200.00	15,996	35.60	53.51	17.92	7,200.00	116,668
G_8 × 80_a	−100.00	–	–	7,200.00	9,507	32.24	46.74	14.50	7,200.00	52,630
G_8 × 80_b	−98.72	–	–	7,200.00	9,797	35.09	52.75	17.66	7,200.00	60,023
G_9 × 90_a	−98.43	–	–	7,200.00	13,802	33.20	50.64	17.44	7,200.00	35,137
G_9 × 90_b	−98.94	–	–	7,200.00	5,997	34.99	51.56	16.57	7,200.00	46,045
G_10 × 100_a	−100.00	–	–	7,200.00	1,273	37.20	57.07	19.87	7,200.00	18,444
G_10 × 100_b	−98.66	–	–	7,200.00	2,425	32.98	47.42	14.45	7,200.00	24,643
Average			–					18.78		

Table 5 Evaluation of the three versions of BRKGA-RSP for Karasan instances

Instance	BRKGA-RSP (V1)		BRKGA-RSP (V2)		BRKGA-RSP (V3)	
	cost (%)	t(s)	cost (%)	t(s)	cost (%)	t(s)
K-1000-200-0.9-a-5	61.51	315.87	61.50	298.38	61.49	281.74
K-1000-200-0.9-b-5	67.53	315.78	67.53	298.44	67.58	281.83
K-1000-200-0.9-a-10	93.44	426.74	93.51	402.29	93.38	380.48
K-1000-200-0.9-b-10	76.82	426.48	76.79	402.33	76.74	380.81
K-1000-200-0.9-a-25	65.07	797.40	65.07	727.43	65.07	711.62
K-1000-200-0.9-b-25	77.93	797.17	77.93	727.26	77.93	711.75
K-1000-200-0.9-a-50	68.09	1,487.05	68.09	1,387.13	68.09	1,339.12
K-1000-200-0.9-b-50	65.91	1,486.61	65.91	1,383.81	65.91	1,337.78
K-1000-200-0.9-a-100	84.21	2,882.54	84.21	2,683.23	84.21	2,593.59
K-1000-200-0.9-b-100	90.00	2,843.23	90.00	2,645.66	90.00	2,553.54
K-1500-200-0.9-a-5	53.01	905.38	53.01	856.77	52.85	809.79
K-1500-200-0.9-b-5	45.17	905.64	45.13	856.63	45.17	809.43
K-1500-200-0.9-a-10	57.52	1,152.59	57.52	1,089.43	57.53	1,030.40
K-1500-200-0.9-b-10	56.39	1,152.47	56.39	1,089.05	56.39	1,030.28
K-1500-200-0.9-a-25	52.88	2,051.09	52.88	1,936.81	52.88	1,842.47
K-1500-200-0.9-b-25	87.57	2,052.06	87.57	1,936.57	87.57	1,843.16
K-1500-200-0.9-a-50	59.26	3,814.90	59.26	3,583.54	59.26	3,463.09
K-1500-200-0.9-b-50	75.51	3,819.99	75.51	3,594.31	75.51	3,469.26
K-1500-200-0.9-a-100	81.25	6,968.95	81.25	6,561.32	81.25	6,304.12
K-1500-200-0.9-b-100	105.88	7,007.78	105.88	6,593.62	105.88	6,331.89
Average	71.25	2,080.49	71.25	1,952.70	71.23	1,875.31

within 7,200 s of running time motivates the study of heuristics for tackling the large instances of *minmax relative regret* RSP. The following experiments evaluate the performance of the heuristics proposed in this paper and compare their results with those of the main heuristics in literature for RSP.

In the third experiment, three versions of BRKGA-RSP are evaluated for Karasan and grid instances. Version V1 has $|TOP| < |BOT|$ (with $|TOP| = 0.1 \times p$ and $|BOT| = 0.2 \times p$), version V2 has $|TOP| = |BOT|$ (with $|TOP| = |BOT| = 0.15 \times p$), and version V3 has $|TOP| > |BOT|$ (with $|TOP| = 0.2 \times p$ and $|BOT| = 0.1 \times p$), where $p = 100$ is the population size. For each version, we performed 20 runs for each instance with different seeds for the pseudorandom number generator. The results are reported in Tables 5 and 6 for Karasan and grid instances, respectively. The first column shows the name of each instance. The second and third columns display respectively the average solution cost and running time (in seconds) provided by BRKGA-RSP (V1). The same data is given for version V2 in columns 4 and 5, and for version V3 in columns 6 and 7, respectively. One can see that the results of the three versions of BRKGA-RSP were similar for both Karasan and grid instances. BRKGA-RSP (V3) had the smallest running times, due to the fact that the size of set TOP is larger than that of the other versions. Therefore, the number of chromosomes evaluated at each iteration of version V3 is smaller than that of the other versions, because chromosomes in set TOP are copied without change.

Table 6 Evaluation of the three versions of BRKGA-RSP for grid instances

Instance	BRKGA-RSP (V1)		BRKGA-RSP (V2)		BRKGA-RSP (V3)	
	cost (%)	t(s)	cost (%)	t(s)	cost (%)	t(s)
G_6x60_a	42.52	22.66	42.48	21.57	42.47	20.53
G_6x60_b	50.09	22.47	50.07	21.41	50.05	20.35
G_7x70_a	58.78	41.89	58.78	39.76	58.78	37.67
G_7x70_b	53.51	41.86	53.51	39.76	53.51	37.66
G_8x80_a	46.69	77.01	46.69	72.95	46.66	69.24
G_8x80_b	52.75	73.06	52.75	69.18	52.75	65.50
G_9x90_a	50.54	124.68	50.47	118.19	50.53	111.81
G_9x90_b	51.86	122.96	51.86	116.45	51.86	110.10
G_10x100_a	57.09	203.41	57.14	193.04	57.09	182.42
G_10x100_b	47.45	196.88	47.45	186.82	47.44	176.70
Average	51.13	92.69	51.12	87.91	51.11	83.20

In the fourth experiment, two versions of SA-RSP are evaluated for Karasan and grid instances. Version V1 has $t_f = 0.1$ as reported in [36], while version V2 has $t_f = 0.01$. The smaller is the value of t_f, the larger is the running time of SA-RSP, and the better are the solution found by this heuristic. The other parameters were set to the same values as described in [36], i.e. $\beta = 0.94$, $\lambda = 25$, and $t_0 = 1$. For each version, we performed 20 runs for each instance with different seeds for the pseudorandom number generator. The results are reported in Tables 7 and 8 for Karasan and grid instances, respectively. The first column shows the name of each instance. The second and third columns display the average solution cost and running times (in seconds) of SA-RSP (V1), respectively. The same data is given for version V2 in columns 4 and 5, respectively. It can be observed that, on average, the results obtained by version V2 are slightly better than those obtained by version V1. However, the average running time of the former is twice that of the latter. This suggests that no significant increase in the solution of SA-RSP can be obtained by decreasing the value of t_f even further.

In the last experiment, we compare the performance of the heuristics AMU [25], PM-RSP, and the best versions of SA-RSP [36] (V2), and BRKGA-RSP (V3). We note that AMU and PM-RSP are deterministic algorithms, while SA-RSP and BRKGA-RSP are stochastic heuristics. The results are reported in Tables 9 and 10 for Karasan and grid instances, respectively. The six instances solved at optimality by CPLEX (V2) before 7,200 s (see Table 3) are indicated by an asterisk. The second and third columns display respectively the average solution cost and running time (in seconds) provided by AMU. The same data is given for SA-RSP [36] in columns 4 and 5, for PM-RSP in columns 6 and 7, and for BRKGA-RSP in columns 8 and 9, respectively. The smallest average solution cost for each instance is displayed in boldface.

Regarding the Karasan instances (Table 9), it can be observed that PM-RSP found better solutions on average (71.99 %) than AMU (76.36 %) and SA-RSP (74.41 %) within an average running time of 12.00 s. The best average result was obtained by BRKGA-RSP (71.23 %). Besides, the latter found solutions as good as or better than the other heuristics in all but instance K-1000-200-0.9-a-5. However, the average running time of the latter was 1,875.31 s. Both PM-RSP and BRKGA-RSP heuristics proposed in this paper found the optimal solutions for the six instances solved by CPLEX (V2) at optimality, while AMU and SA-RSP found optimal solution only for four and three of these instances, respectively. Regarding the grid

Table 7 Evaluation of the two versions of SA-RSP for Karasan instances

Instance	SA-RSP (V1)		SA-RSP (V2)	
	cost (%)	t(s)	cost (%)	t(s)
K-1000-200-0.9-a-5	64.07	2.14	64.07	4.15
K-1000-200-0.9-b-5	78.12	2.12	78.12	3.90
K-1000-200-0.9-a-10	96.94	2.73	96.94	5.18
K-1000-200-0.9-b-10	82.56	2.98	82.56	6.48
K-1000-200-0.9-a-25	66.90	6.18	66.90	18.99
K-1000-200-0.9-b-25	78.54	5.37	78.54	12.47
K-1000-200-0.9-a-50	68.09	8.39	68.09	16.51
K-1000-200-0.9-b-50	65.91	8.33	65.91	16.38
K-1000-200-0.9-a-100	84.21	11.21	84.21	16.92
K-1000-200-0.9-b-100	91.50	11.22	90.75	16.98
K-1500-200-0.9-a-5	56.90	3.92	56.90	6.46
K-1500-200-0.9-b-5	48.61	3.91	48.61	6.36
K-1500-200-0.9-a-10	60.52	5.01	60.52	9.56
K-1500-200-0.9-b-10	59.38	5.09	59.38	9.47
K-1500-200-0.9-a-25	55.84	9.15	55.84	27.80
K-1500-200-0.9-b-25	94.90	10.13	94.90	30.34
K-1500-200-0.9-a-50	63.04	16.37	62.41	46.32
K-1500-200-0.9-b-50	87.34	20.77	84.08	56.14
K-1500-200-0.9-a-100	83.15	31.42	82.06	50.48
K-1500-200-0.9-b-100	110.25	21.97	107.35	38.39
Average	74.84	9.42	74.41	19.96

Table 8 Evaluation of the two versions of SA-RSP for grid instances

Instance	SA-RSP (V1)		SA-RSP (V2)	
	cost (%)	t(s)	cost (%)	t(s)
G_6 × 60_a	49.45	2.84	49.45	4.98
G_6 × 60_b	54.09	2.90	54.09	5.18
G_7 × 70_a	61.00	5.45	61.00	9.60
G_7 × 70_b	57.42	5.31	57.42	9.42
G_8 × 80_a	49.46	9.38	49.46	16.97
G_8 × 80_b	54.31	9.26	54.31	16.61
G_9 × 90_a	57.83	17.23	57.83	34.78
G_9 × 90_b	56.04	15.32	56.04	28.95
G_10 × 100_a	69.39	26.35	69.39	53.94
G_10 × 100_b	54.10	29.74	54.10	65.48
Average	56.31	12.38	56.31	24.59

instances (Table 10), it can be observed that PM-RSP found again better solutions on average (51.92 %) than AMU (53.42 %) and SA-RSP (56.31 %) within an average running time of 0.34 s. The best average result was again obtained by BRKGA-RSP (51.11 %). Besides, the latter found solutions as good as or better than the other heuristics in all grid instances. However, the average running time of the latter was 83.20 s.

Table 9 Comparison between the heuristics proposed in this paper and the main heuristics in the literature of RSP for Karasan instances

Instance	AMU [25]		SA-RSP [36] (V2)		PM-RSP		BRKGA-RSP (V3)	
	cost (%)	t(s)	cost (%)	t(s)	cost (%)	t(s)	cost (%)	t(s)
K-1000-200-0.9-a-5	64.07	0.01	64.07	4.15	**61.36**	2.80	61.49	281.74
K-1000-200-0.9-b-5	74.13	0.01	78.12	3.90	68.86	2.83	**67.58**	281.83
K-1000-200-0.9-a-10	96.94	0.01	96.94	5.18	99.78	3.50	**93.38**	380.48
K-1000-200-0.9-b-10	82.56	0.01	82.56	6.48	77.23	3.46	**76.74**	380.81
K-1000-200-0.9-a-25	66.90	0.01	66.90	18.99	67.36	5.64	**65.07**	711.62
K-1000-200-0.9-b-25	78.57	0.01	78.54	12.47	80.14	5.64	**77.93**	711.75
K-1000-200-0.9-a-50*	**68.09**	0.04	**68.09**	16.51	**68.09**	8.19	**68.09**	1,339.12
K-1000-200-0.9-b-50*	**65.91**	0.03	**65.91**	16.38	**65.91**	8.17	**65.91**	1,337.78
K-1000-200-0.9-a-100*	**84.21**	0.05	**84.21**	16.92	**84.21**	13.15	**84.21**	2,593.59
K-1000-200-0.9-b-100*	100.00	0.05	90.75	16.98	**90.00**	12.98	**90.00**	2,553.54
K-1500-200-0.9-a-5	56.68	0.01	56.90	6.46	53.31	8.56	**52.85**	809.79
K-1500-200-0.9-b-5	47.87	0.01	48.61	6.36	46.74	8.54	**45.17**	809.43
K-1500-200-0.9-a-10	60.52	0.02	60.52	9.56	58.19	10.02	**57.53**	1,030.40
K-1500-200-0.9-b-10	59.20	0.02	59.38	9.47	**56.39**	10.03	**56.39**	1,030.28
K-1500-200-0.9-a-25	53.88	0.04	55.84	27.80	**52.88**	14.79	**52.88**	1,842.47
K-1500-200-0.9-b-25	89.34	0.04	94.90	30.34	**87.57**	14.74	**87.57**	1,843.16
K-1500-200-0.9-a-50*	63.46	0.05	62.41	46.32	**59.26**	20.76	**59.26**	3,463.09
K-1500-200-0.9-b-50	79.59	0.06	84.08	56.14	**75.51**	20.62	**75.51**	3,469.26
K-1500-200-0.9-a-100*	**81.25**	0.1	82.06	50.48	**81.25**	32.85	**81.25**	6,304.12
K-1500-200-0.9-b-100	**105.88**	0.11	107.35	38.39	**105.88**	32.67	**105.88**	6,331.89
Average	76.36	0.03	74.41	19.96	71.99	12.00	**71.23**	1,875.31

Table 10 Comparison between the heuristics proposed in this paper and the main heuristics in the literature of RSP for grid instances

Instance	AMU [25]		SA-RSP [36] (V2)		PM-RSP		BRKGA-RSP (V3)	
	cost (%)	t(s)	cost (%)	t(s)	cost (%)	t(s)	cost (%)	t(s)
G_6x60_a	44.24	0.01	49.45	4.98	43.03	0.11	**42.47**	20.53
G_6x60_b	54.09	0.01	54.09	5.18	53.83	0.10	**50.05**	20.35
G_7x70_a	61.00	0.01	61.00	9.60	59.68	0.18	**58.78**	37.67
G_7x70_b	54.95	0.01	57.42	9.42	53.96	0.16	**53.51**	37.66
G_8x80_a	49.46	0.01	49.46	16.97	48.02	0.32	**46.66**	69.24
G_8x80_b	54.31	0.01	54.31	16.61	**52.75**	0.26	**52.75**	65.50
G_9x90_a	51.38	0.01	57.83	34.78	50.77	0.47	**50.53**	111.81
G_9x90_b	56.04	0.01	56.04	28.95	52.32	0.45	**51.86**	110.10
G_10x100_a	59.22	0.01	69.39	53.94	57.30	0.73	**57.09**	182.42
G_10x100_b	49.50	0.01	54.10	65.48	47.50	0.66	**47.44**	176.70
Average	53.42	0.01	56.31	24.59	51.92	0.34	**51.11**	83.20

These results suggest that PM-RSP has the best cost-benefit ratio among of the heuristics studied in this paper, as it can find solutions almost as good as those of BRKGA-RSP, with considerably less computational effort. The former can find good solutions in considerable less time than the latter, because instead of relying on the evolutionary processes to identify the good solutions, the former relies on the AM based heuristic to guide search for good solutions.

6 Concluding remarks

In this work, we proposed the first integer linear programming formulation for *minmax relative regret*, as well as valid inequalities and metaheuristics. Experiments were performed on the classic instances based on Karasan graphs, and on new instances based on *grid* graphs. The CPLEX branch-and-bound algorithm based on this formulation found optimal solutions for most of the small Karasan and grid instances with up to 200 nodes. Besides, it also provided integer feasible solutions as good as or better than the best algorithms in the literature of RSP for the large Karasan and grid instances with up to 1,500 nodes. The grid instances proposed in this paper showed up to be harder to solve than the Karasan instances found in the literature.

In addition, we developed heuristics with emphasis on providing efficient and scalable methods for solving the *minmax relative regret* RSP. The biased random-key genetic algorithm was faster than and found solutions as good as those of the CPLEX branch-and-bound algorithm, on average. Moreover the Pilot method found solutions almost as good as those of the latter but in considerably smaller running times.

The linearization method and the valid inequalities applied to the *minmax relative regret* objective function of RSP can also be applied to other *minmax relative regret* robust optimization problems. Besides, both the Pilot method and the genetic algorithm frameworks proposed for RSP can be extended for other robust optimization problems by simply exchanging the problem dependent guiding heuristic (in the case of the Pilot method) and the decoding heuristic (in the case of the BRKGA).

Acknowledgments This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq), the Foundation for Support of Research of the State of Minas Gerais, Brazil (FAPEMIG), and Coordination for the Improvement of Higher Education Personnel, Brazil (CAPES).

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Min-max and min-max regret versions of combinatorial optimization problems: a survey. Eur. J. Oper. Res. **197**, 427–438 (2009)
2. Averbakh, I.: Computing and minimizing the relative regret in combinatorial optimization with interval data. Discrete Optim. **2**, 273–287 (2005)
3. Averbakh, I., Lebedev, V.: Interval data minmax regret network optimization problems. Discrete Appl. Math. **138**, 289–301 (2004)
4. Bean, J.C.: Genetic algorithms and random keys for sequencing and optmazion. ORSA J. Comput. **2**, 154–160 (1994)
5. Bellman, R.: On a routing problem. Q. Appl. Math. **16**, 87–90 (1958)
6. Ben-Tal, A., Nemirovski, A.: Robust optimization—methodology and applications. Math. Program. **92**, 453–480 (2002)
7. Bertsekas, D.P., Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. Math. Oper. Res. **16**, 580–595 (1991)
8. Bisschop, J.: AIMMS—Optimization Modeling. Integer Linear Programming Tricks. Paragon Decision Technology B.V., Haarlem (2005)

9. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. Elsevier, Amsterdam (1976)
10. Candia-Véjar, A., Álvarez-Miranda, E., Maculan, N.: Minmax regret combinatorial optimization problems: an algorithmic perspective. RAIRO Oper. Res. **45**, 101–129 (2011)
11. Catanzaro, D., Labbé, M., Salazar-Neumann, M.: Reduction approaches for robust shortest path problems. Comput. Oper. Res. **38**, 1610–1619 (2011)
12. Conde, E.: On a constant factor approximation for minmax regret problems using a symmetry point scenario. Eur. J. Oper. Res. **219**, 452–457 (2012)
13. Desrochers, M., Laporte, G.: Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. Operat. Res. Lett. **10**, 27–36 (1991)
14. Dijkstra, E.W.: A note on two problems in connection with graphs. Numer. Math. **1**, 269–271 (1959)
15. Duin, C., Voss, S.: The Pilot Method: A strategy for heuristic repetition with application to the Steiner problem in Graphs. Networks **34**, 181–191 (1999)
16. Escoffier, B., Monnot, J., Spanjaard, O.: Some tractable instances of interval data minmax regret problems. Oper. Res. Lett. **36**, 424–429 (2008)
17. Gabrel, V., Murat, C., Wu, L.: New models for the robust shortest path problem: complexity, resolution and generalization. Ann. Oper. Res. **207**, 97–120 (2013)
18. Gallo, G., Pallottino, S.: Shorthest path methods: a unifying approach. Math. Progam. Study **26**, 38–64 (1986)
19. Gonçalves, J.F., Resende, M.G.C.: Biased random-key genetic algorithms for combinatorial optimization. J. Heuristics **17**, 487–525 (2011)
20. Gonçalves, J.F., de Magalhaes Mendes, J.J., Resende, M.G.C.: A genetic algorithm for the resource constrained multi-project scheduling problem. Eur. J. Oper. Res. **189**, 1171–1190 (2008)
21. Gonçalves, J.F., Rezende, M.G.C.: Biased random-key genetic algorithms for combinatorial optmazation. J. Heuristics **17**, 487–525 (2010)
22. Karasan, O.E., Yaman, H., Ç. Pinar, M.: The Robust Shortest Path Problem with Interval Data. Technical report, Bilkent University, Department of, Industrial Engineering (2001)
23. Kasperski, A., Kobylański, P., Kulej, M., Zieliński, P.: Minimizing Maximal Regret in Discrete Optimization Problems with Interval Data, pp. 193–208. Akademicka Oficyna Wydawnicza EXIT, Warszawa (2005)
24. Kasperski, A., Zieliński, P.: The robust shortest path problem in series-parallel multidigraphs with interval data. Oper. Res. Lett. **34**, 69–76 (2006)
25. Kasperski, A., Zieliński, P.: On the existence of an FPTAS for minmax regret combinatorial optimization with interval data. Oper. Res. Lett. **35**, 525–532 (2007)
26. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
27. Kouvelis, P., Yu, G.: Robust discrete optimization and its applications. Kluver, Boston (1997)
28. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**, 3–30 (1998)
29. Mendes, J.J.M., Gonçalves, J.F., Resende, M.G.C.: A random key based genetic algorithm for the resource constrained project scheduling problem. Comput. Oper. Res. **36**, 92–109 (2009)
30. Miller, C., Tucker, A., Zemlin, R.: Integer programming formulations and traveling salesman problems. J. ACM **7**, 326–329 (1960)
31. Montemanni, R., Gambardella, L.M.: A branch and bound algorithm for the robust spanning tree problem with interval data. Eur. J. Oper. Res. **161**, 771–779 (2005)
32. Montemanni, R., Gambardella, L.M.: The robust shortest path problem with interval data via Benders decomposition. 40R **3**, 315–328 (2005)
33. Montemanni, R., Gambardella, L.M., Donati, A.V.: A branch and bound algorithm for the robust shortest path problem with interval data. Oper. Res. Lett. **32**, 225–232 (2004)
34. Nie, Y., Wu, X.: Shortest path problem considering on-time arrival probability. Transp. Res. Part B **43**, 597–613 (2009)
35. Noronha, T.F., Rezende, M.G.C., Ribeiro, C.C.: A biased random-key genetic algorithm for routing and wavelength assignment. J. Global Optim. **50**, 503–518 (2011)
36. Pérez, F., Astudillo, C.A., Bardeen, M., Candia-Véjar, A.: A simulated annealing approach for the minmax regret path problem. In: Proceedings of the Congresso Latino Americano de Investigación Operativa (CLAIO)—Simpósio Brasileiro de Pesquisa Operacional (SBPO) 2012. Rio de Janeiro, Brazil (2012)
37. Resende, M.G.C., Toso, R.F., Gonçalves, J.F., Silva, R.M.A.: A biased random-key genetic algorithm for the steiner triple covering problem. Optim. Lett. **6**, 605–619 (2012)
38. Santos, A., Duhamel, C., Aloise, D.: Modeling the mobile oil recovery problem as a multiobjective vehicle routing problem. Model. Comput. Optim. Inf. Syst. Manag. Sci. **14**, 283–292 (2008)
39. Spall, J.C.: Introduction to Stochastic Search and Optimization. Wiley, New York (2003)

40. Spears, W., DeJong, K.: On the virtues of parameterized uniform crossover. In: Belew, R., Booker, L. (eds.) Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 230–236. San Mateo, Italy (1991)
41. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Trans/. Syst. Man Cybern. **2**, 109–125 (1981)
42. Toso, R.F., Rezende, M.G.C.: A C++ application programmming interface for biased random-key algorithms (2011). http://www2.research.att.com/mgcr/doc/brkgaAPI.pdf (Online; Accessed in 20/03/2014)
43. Voss, S., Fink, A., Duin, C.: Looking ahead with the Pilot method. Ann. Oper. Res. **136**, 285–302 (2005)