A Heuristic Approach for the Stochastic Steiner Tree Problem

Pedro Hokama*, Mário C. San Felice**,
Evandro C. Bracht***, and Fábio L. Usberti†

University of Campinas - UNICAMP, Campinas SP 13083-852, Brazil,
http://www.ic.unicamp.br/~fusberti

Abstract. This paper addresses the two-stage stochastic Steiner tree problem (SSTP), which is one of the Steiner problems that belongs to the 11th DIMACS challenge. In the SSTP there is a finite number of scenarios in which the terminal set and the edge costs are subject to uncertainty. In the first stage, we have the probabilistic information in terms of possible scenarios. A scenario specifies a terminal set, second stage edge costs, and a probability for it to be realized. The objective is to select edges to be purchased in the first stage, while minimizing the expected cost of the full solution.
In this paper we propose a biased random-key genetic algorithm (BRKGA) for the SSTP. A constructive heuristic, based on reduced cost information of the edges, is used to populate the initial set of solutions for the BRKGA. The proposed metaheuristic was extensively tested with instances from literature and from the DIMACS benchmark, which includes large scale instances, with 20 times more scenarios than previously tested instances. Results show that the BRKGA obtained near optimal solutions to the instances from literature and achieved considerable cost reductions for the DIMACS instances.

Keywords: Steiner Tree, Stochastic Optimization, Genetic Algorithms, BRKGA, DIMACS challenge.

1 Introduction

The classical Steiner tree problem (STP) in graphs is a combinatorial optimization problem with several applications, including: network design problems (e.g., communication and power systems) [1984], wire routing in VLSI circuits [1990] and the study of phylogenetic trees [1967]. The objective of the STP is to find a minimum cost network spanning a given subset of nodes (*terminals*).

* hokama@ic.unicamp.br. Supported by FAPESP grant 2011/13382-3
** felice@ic.unicamp.br. Partially supported by FAPESP grant 2009/15535-1.
*** evandro@ic.unicamp.br. Partially supported by FAPESP grant 03/13815-0 and State University of Mato Grosso do Sul - UEMS
† fusberti@ic.unicamp.br. Partially supported by CNPq grant 483137/2013-8

In practice, network planners are often faced with uncertainty with respect to the input data. Therefore, solving an instance in which the knowledge of the input is not certain might provide solutions whose cost deviate substantially from the optimum value.

Stochastic optimization is a field of investigation that takes into account uncertainties arising from real-life optimization problems. In the stochastic Steiner tree problem (SSTP) the terminal nodes are only known probabilistically. Thus, instead of buying edges to connect a given terminal set, edges are bought in a first stage considering a set of possible future scenarios, one of which will eventually arise. Each scenario is characterized by its terminal set, second stage edge costs and an occurrence probability. In a second stage, edges at an inflated cost are bought to complete a Steiner tree with the first stage edges for the scenario's terminal set. The expected cost of a solution is the sum of first stage edge costs plus the weighted sum of second stage edge costs of all scenarios. The weights correspond to the scenarios probabilities.

Figs. 1-2 depict an instance of the SSTP with ten nodes and five scenarios, and its corresponding optimum solution, which includes the first and second stage edges.

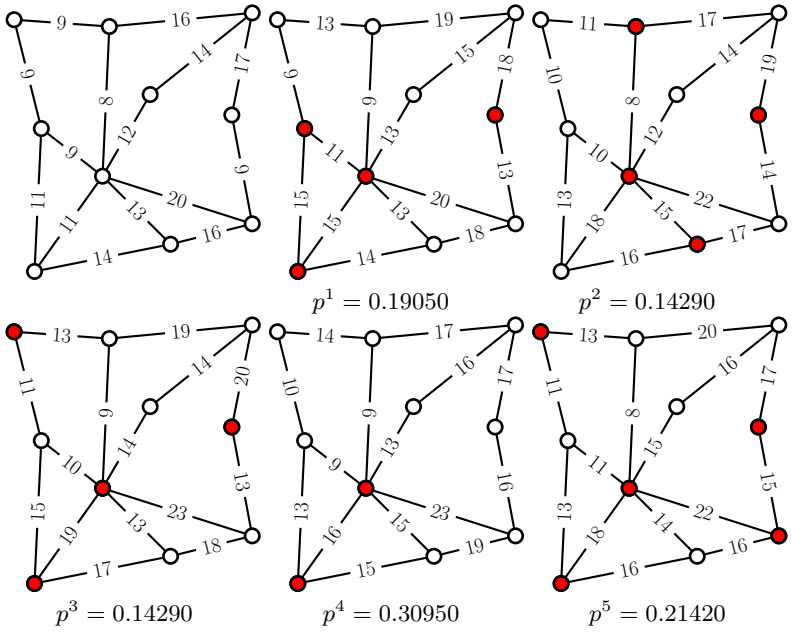

Fig. 1. SSTP instance with five scenarios (terminals in red).

Gupta et al. [2007] investigated approximation algorithms for the SSTP, proving a constant factor approximation for the SSTP when the second stage costs

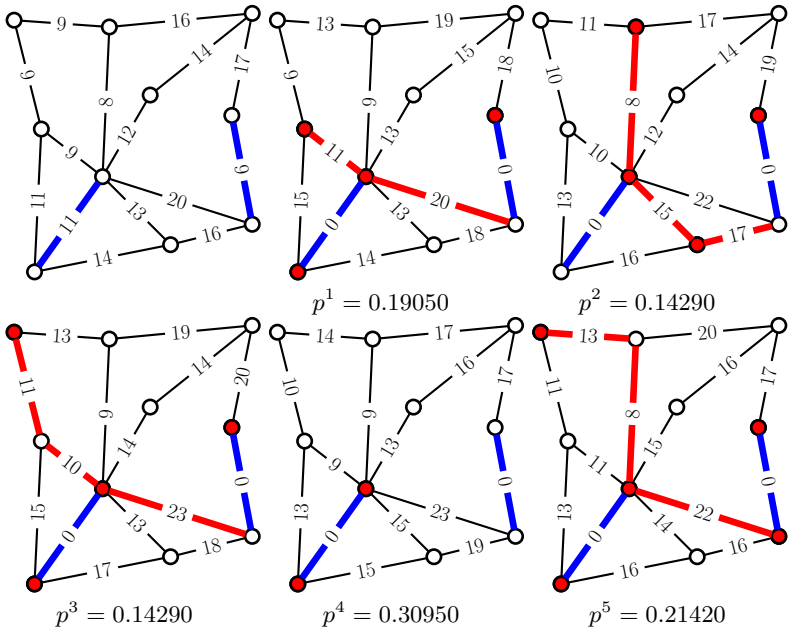

Fig. 2. Solution of the first (blue) and second (red) stages.

are determined by a fixed inflation factor. The algorithm is based on a primal-dual scheme, guided by a relaxed integer linear programming (ILP) solution. Gupta et al. [2007] have also shown that the general case (non-fixed inflation) is as hard as Label Cover ($\Omega(2^{\log^{1-\epsilon} n})$-hard). Shmoys and Swamy [2006] presented a 4-approximation for the SSTP using cost-sharing properties.

Bomze et al. [2010] proposed a two-stage branch-and-cut algorithm which uses integer L-shaped cuts. The authors present the first computational studies for the SSTP, showing optimal solutions for instances up to 50 scenarios and 274 edges.

Our Contributions. In this paper, we propose an effective biased random key genetic algorithm (BRKGA) for the SSTP. Our methodology obtains low cost solutions for large scale instances with 20 times more scenarios than previous instances from literature. This improvement makes our methodology a viable candidate to solve real-life network design problems.

The BRKGA initial population of feasible solutions is generated by a constructive heuristic, called *reduced-cost heuristic*, which uses reduced-cost information of the edges to iteratively select first stage edges. The second stage edges are chosen by solving each scenario with a minimum spanning tree algorithm. To intensify the search, the BRKGA explores the population diversity by applying selective pressure under an evolutive environment. A set of instances from literature (Bomze et al. [2010]) were used in the computational experiments to

compare our results with known optimal solutions. Moreover, our methodology effectively solved the benchmark set of instances from the 11th DIMACS challenge[1].

This paper is organized as follows. In Section 2 we briefly discuss the STP. Section 3 explains the SSTP and the branch-and-cut algorithm from Bomze et al. [2010]. Section 3 also details the proposed methodologies (reduced-cost heuristic and BRKGA). Section 4 shows the computational experiments with instances from literature and from the DIMACS benchmark. The final remarks are presented in Section 5.

2 The Steiner Tree Problem (STP)

In this section we formally describe the Steiner Tree problem and two algorithms to solve it. The first algorithm is a Branch-and-Cut algorithm that solve the STP to optimality. Unfortunately the exact approach can not be applied to large instances in a reasonable running time, therefore we have the second algorithm which is a fast 2-approximation heuristic based on minimum spanning tree.

The STP can be described as follows. Let $I = (G, C, R)$ be an instance for the STP, where $G = (V, E)$ is a graph with $|V| = n$ and $|E| = m$, $C = \{c_e : e \in E\}$ is the edges cost set, and $R \subseteq V$ is the terminal set. A set of edges T, corresponding to a tree in G that connects the nodes in R, is a the solution for this problem. Thus, the STP is the problem of determining a subset of edges $T \subseteq E$, where T spans R, such that the cost $\sum_{e \in T} c_e$ is minimized.

2.1 Branch-and-Cut Algorithm

This algorithm is based on the ILP formulation presented by Bomze et al. [2010]. This formulation uses a directed graph, so for each original undirected edge $e = \{i, j\} \in E$, we create arcs (i, j) and (j, i). Let A be the set of all arcs, $(i, j) \in A$ be a directed arc from i to j, and z_{ij} be a variable that indicates if arc (i, j) is in the solution. Defining $r \in R$ as an arbitrary terminal node, and $V_r = V \setminus \{r\}$, we consider the following formulation:

$$\min \sum_{e=\{i,j\} \in E} (z_{ij} + z_{ji}) c_e$$

$$s.t.$$
$$z(\delta^-(S)) \geq 1, \forall S \subseteq V_r,\ S \cap R \neq \emptyset$$

$$z_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A.$$

The number of constraints in this formulation is exponential, thus we apply the branch-and-cut strategy. First we consider a relaxed form of the problem by removing the constraints, both for connectivity and integrality. Then, for each solution found for the relaxed formulation, we search for violated constraints.

[1] http://dimacs11.cs.princeton.edu/

To find a violated constraint we use the following algorithm. Given a solution z for the relaxed formulation, we associate a capacity of value z_{ij} to each arc (i, j). Then, for each terminal $t \in R$ we solve a minimum cut problem, and let $S \subseteq V_r$ be the minimum cut between t and the root r. If $z(\delta^-(S)) < 1$, we remove this solution adding the constraint $z(\delta^-(S)) \geq 1$ to the formulation, and the new problem is optimized. If there are no more violated cuts in the current node of the branch-and-bound tree, the algorithm branches and the process is repeated in the following nodes.

This branch-and-cut algorithm finds the optimal solution to the STP, but in reasonable time, can be applied only to moderate-size instances.

2.2 MST-Approx

A fast algorithm to solve the STP is essential to solve the SSTP, because only to find the cost of a SSTP solution, it is necessary to solve several STP instances, one for each scenario. Thus, we use the minimum spanning tree 2-approximation algorithm to the STP (Vazirani [2003]), which we call MST-Approx.

The first step is to compute the shortest path between all pairs of terminals. Let $P_{ij} \subseteq E$ be the shortest path between terminals i and j. Then we create a new complete graph $G' = \{R, E'\}$ where each (i, j) from E' represents the path P_{ij} and has cost $c'_{ij} = \sum_{e \in P_{ij}} c_e$. The graph G' is called a metric-completion of G restricted to R. The minimum paths represented in the edges of the graph G' are not necessarily disjoint in the original graph G.

The second step is to compute a minimum spanning tree T' of G'. Then, a Steiner tree T is built for G based on T'. For each edge $(i, j) \in T'$ we add to T all edges in P_{ij}, except for repeated edges and for those that may form cycles. All steps of this algorithm can be performed in polynomial time.

3 Stochastic Steiner Tree Problem

In this section we describe the Two-Stage Stochastic Steiner Tree problem (SSTP) and show some approaches to solve it.

We use a similar notation from Bomze et al. [2010]. Let $I = (G, C, P, \mathcal{Q}, r, \mathcal{R})$ be an instance for the SSTP, where $G = (V, E)$ is a graph with $|V| = n$ and $|E| = m$, $C = \{c_e : e \in E\}$ is the edges cost set for the first stage, and $P = \{p^1, \ldots, p^k\}$ is the set of occurrence probability for the scenarios. Also, $\mathcal{Q} = \{Q^1, \ldots, Q^k\}$, where $Q^l = \{q^l_e : e \in E\}$ is the edge cost set in the l-th scenario, and r is the root node. Defining $V_r = V \setminus \{r\}$, we have $\mathcal{R} = \{R^1, \ldots, R^k\}$, where $R^l \subseteq V_r$ is the terminal set in the l-th scenario.

We denote by E_0 the set of edges purchased in the first stage, and by E_l the set of additional edges purchased under scenario l. Thus, the SSTP is the problem of determining a subset of edges $E_0 \subseteq E$ to be purchased in the first stage, and the subsets of edges E_l, for $l = 1, \ldots, k$, where $E_0 \cup E_l$ spans $R_l \cup \{r\}$, such that the expected cost defined as $\sum_{e \in E_0} c_e + \sum_{l=1}^{k} p_l \sum_{e \in E_l} q^l_e$ is minimized.

Bomze *et al.* [2010] presents a Two-Stage Branch-and-Cut framework for the Stochastic Steiner Tree Problem, that consists of a semi-directed ILP model, that is stronger than the undirected ILP model proposed by Gupta *et al.* [2007]. To the best of our knowledge, Bomze *et al.* [2010] were the first to report experimental results about the SSTP.

3.1 Buy-None Heuristic

The first and most trivial heuristic for the SSTP, called Buy-None, consists of letting all edges to be bought in the second stage of the problem. To solve each scenario of the second stage, it is necessary to solve a STP, which can be done optimally or approximately.

When the second stage scenarios are solved optimally, we have that the gap between this solution cost and the SSTP optimal cost is upper bounded by the highest inflation α of the edges. More formally, let I be an instance for the SSTP and $OPT(I)$ be the cost of an optimal solution for I. Also, let $buyNoneExact(I)$ be the cost of a solution returned by Buy-None to solve I, with the second stage scenarios solved by an exact STP algorithm. We have that $buyNoneExact(I) \leq (1+\alpha)OPT(I)$.

Unfortunately, solving the second stage optimally is practical only for moderate-size instances. Therefore, it is necessary to use a fast STP algorithm, like the MST-Approx. Also, using the Buy-None approach with a β-approximation for solving the STP, we obtain a $\beta(1+\alpha)$-approximation. Let $buyNoneMST(I)$ be the cost of a solution returned by Buy-None to solve I, with the second stage scenarios solved by MST-Approx. Since MST-Approx is a 2-approximation for STP, we have that $buyNoneMST(I) \leq 2buyNoneExact(I) \leq 2(1+\alpha)OPT(I)$.

3.2 Reduced-Cost Heuristic

Our second heuristic, called Reduced-Cost, consists of estimating the effects of adding a new edge to the first stage of a partial solution, and actually adding only those edges which reduce the estimated cost of this solution, i.e. the edges with negative reduced cost. The algorithm repeats this process until no more profitable edges are found or a time limit is reached.

To estimate the cost of a partial solution, for each scenario we set the cost of edges added in the first stage to zero, and use the MST-Approx algorithm to find a Steiner Tree to that scenario. Let I be an instance for the SSTP, and $S \subseteq E$ be a set of edges selected for the first stage of a solution. We define $MC(I,S)$ as the cost to solve I, when S is bought in the first stage and the second stage scenarios are solved using the MST-Approx algorithm.

The Reduced-Cost heuristic first estimates the cost of $MC(I,\emptyset)$, i.e. a solution in which no edges are bought in the first stage. This step is exactly the solution of Buy-None heuristic. Then, the algorithm estimates the reduced cost $rc(I,S,e) = MC(I,S\cup\{e\}) - MC(I,S)$ for each edge $e \in E$, in an arbitrary order. When an edge with negative reduced cost is found, it is added to S. The

algorithm also stops this process if, to a given partial solution S, the reduced cost $rc(I, S, e) > 0$ for all edges $e \in E$, or it reaches a time limit.

The algorithm then verifies if removing any of the edges previously selected reduces the cost of the current solution, i.e. for all edges $e \in S$, if $MC(I, S \backslash \{e\}) - MC(I, S) < 0$, then the algorithm removes e from S. This process is repeated until, after running over all edges in S no modification was done. The result is a set S of edges to be bought on first stage.

3.3 Biased Random-Key Genetic Algorithm (BRKGA)

The BRKGA, presented by Gonçalves [2011], is a general search metaheuristic based in genetic algorithms, where a population of individuals evolves through the Darwinian principle of survival of the fittest. Each individual of this population is a chromosome that is composed by a set of uniformly drawn random keys (alleles) that represent a solution with a fitness.

Two key features distinguish BRKGA from traditional genetic algorithms [1989]:

1. A chromosome encoding that adopts a vector with m uniformly drawn random keys (alleles) over the interval $[0, 1]$, where m depends on the instance of the optimization problem;
2. For intensification, it uses parameterized uniform crossover [1991]. For diversification, it substitutes the mutation operator on chromosomes with newly introduced mutants, defined as m-long vectors of random keys.

The traditional BRKGA initializes the population with p randomly generated chromosomes, each having m random keys. The population evolves through several generations, each one composed of the following steps:

1. Evaluate the fitness for each individual.
2. Produce the next generation population:
   (a) Select the best p_e individuals from the current population, as they will form the *elite set*.
   (b) Generate p_m new mutants.
   (c) Produce new chromosomes through uniform crossovers between two individuals from the current generation, one from the elite set and another from the non-elite set. Each gene has probability ρ_e of being inherited from the elite parent.

The population size p, elite set size p_e, number of mutants p_m and the elite crossover probability ρ_e are the BRKGA parameters.

To use this metaheuristic as a framework to an optimization problem we need the following steps: define the number of genes in a chromosome, define a decoder which maps a chromosome into a solution, and define the fitness value of the chromosome, which will measure the quality of the solution.

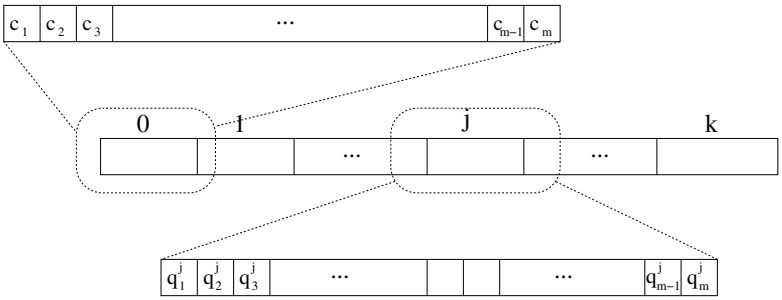

Fig. 3. The chromosome structure.

Chromosome. The chromosome, for an instance $\mathcal{I}$ of SSTP with m edges and k scenarios, is a vector of size $(k+1) \cdot m$, where the first m positions are related to the first-stage edges, and the $k \cdot m$ subsequent positions correspond to the edges of all k scenarios. Figure 3 illustrates a chromosome decomposition.

Initial Population In the traditional BRKGA the initial population are composed of p chromosomes made of uniformly drawn random keys (alleles), however, is possible to manipulate this first population by introducing some chromosomes that represent some solutions obtained from other methods. It can be done with the aim to include some previous knowledge about the problem, that can help in the search for better solutions. Usually this strategy reduce the BRKGA convergence time. In our approach we introduce, in the first population, a chromosome that represent the Buy-None Heuristic solution and a chromosome that represent the best solution found by the Reduced-Cost Heuristic.

Decoder. The role of the decoder is to translate a given chromosome to a feasible solution. Let X be a chromosome. Initially our decoder look at the first m positions of X and, if $X[e] \leq 0.05$ then include the edge e in the first stage. For each edge included in the first stage, its value in all k scenarios are set equal to zero.

For the decoding of the second stage, the chromosome keys of a given scenario l are used as perturbations of the corresponding edge costs. The perturbed cost of an edge e will be $q'^l_e = (0.5 + q^l_e) \cdot b^l_e$, where b^l_e is the introduced perturbation. If an edge e was chosen for the first stage, then $q'^l_e = 0$. With these perturbed costs, each scenario is solved using the MST-Approx.

Figure 4(a-c) illustrates the advantage of using perturbed costs in the decoding process. Consider a single scenario, where the edge set is $E = \{a, b, c, d, e, f\}$, with edge costs $C = \{9, 5, 9, 5, 5, 9\}$. Figure 4.b shows a sub-optimal solution, with total cost 18, returned by the MST-Approx using the original edge costs. Now, assuming a chromosome $X = [0.7, 0.4, 0.5, 0.5, 0.3, 0.6]$ and computing the edge cost using $q'^l_e = (0.5 + q^l_e) \cdot b^l_e$, the new edge costs will be $C' =$

$\{10.8, 4.5, 9, 5, 4, 9, 9\}$. With these costs the MST-Approx returns the optimal solution (Figure 4.c) with real cost 15.

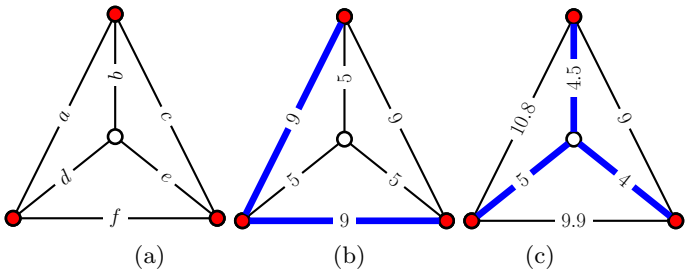

Fig. 4. Example of improved solution.

Combining the set of edges used in each scenario with the edges selected in the first stage results in a complete SSTP solution. Given we have a minimization problem, the solution's fitness is determined as the inverse of its expected cost.

4 Results

This section shows our computational experiments with the BRKGA using two benchmark sets of instances. The first benchmark contains 24 instances (5-50 scenarios, 80-274 edges, $\alpha = 0.5$), extracted from the work of Bomze et al. [2010], for which optimal solutions are known for all except one instance. The second benchmark is part of the 11th DIMACS Implementation Challenge [2014]. This benchmark is divided into four types of instances (K, P, Lin and Wrp), totaling 560 instances (5-1000 scenarios, 64-613 edges, $\alpha = 1.0$). To the best of our knowledge, this is the first work reporting results for this benchmark.

Our computational experiments were executed on an Intel Xeon CPU E3-1230 V2, 3.30GHz, with 32 GB RAM, under Ubuntu 14.04. All codes were implemented in C++ using the Lemon Graph Library [2014] as the framework for graph data structures. A one hour limit of computational time was adopted as stopping criteria, which includes a 30 minutes limit for the reduced cost heuristic.

Table 4 shows the parameter settings for the BRKGA. It should be noticed that the BRKGA population size is within the range of $[10, 100]$ individuals, and varies with respect to the instance size. All the remaining parameters values were suitably adjusted after exhaustive experimentation.

4.1 Results for Benchmark 1

Table 4.1 shows results for the instances previously solved by an exact algorithm from Bomze et al. [2010]. In their paper, only one instance (lin05_160_269, $k =$

Table 1. BRKGA parameters setting.

population size	$p = \max\left[10, \min\left[100, \left\lceil \frac{\gamma}{k\lvert E\rvert} \right\rceil\right]\right]$
population size multiplier	$\gamma = 2,812,500$
elite set size	$p_e = \lceil 0.1p \rceil$
number of mutants	$p_m = \lceil 0.2p \rceil$
elite crossover probability	$\rho_e = 0.825$

20) remained without an optimal solution, given the time limit of two hours. In this case, the authors reported the best feasible solution obtained by their branch-and-cut algorithm, which is also replicated in Table 4.1.

The BRKGA has shown a good performance for instances with known optimums, achieving small optimality gaps, with an average of 0.47%. For the instance with unknown optimum, the BRKGA has found a solution with 4.36% lower cost than the previous best. Within the one hour limit, the BRKGA required in average 943 seconds to deliver its best solution, which indicates an overall fast convergence of our approach.

4.2 Results for Benchmark 2

Table 4.2 shows summarized results[2] for the DIMACS benchmark, displaying average values for each one of the four instance types and for each considered number of scenarios. The exhibited values concern BRKGA solutions with respect to their average cost reduction from reference solutions and their average execution time. Without the knowledge of the optimal solutions, we have used the *buy_none* solutions as references to determine the cost reduction obtained by the BRKGA. This comparison informs how much would a decision maker gain by anticipating future demands in his network, thus buying cheaper edges in the present.

The results show that our methodology considerably reduces the costs of instances from diverse types and number of scenarios. The overall average cost reduction achieved by BRKGA, compared to the *buy_none* solutions, was 3.42%. It is noticeable that the cost reduction decreases when more scenarios are present. Two possible explanations for this are: (*i*) the instances are harder to solve, thus longer execution times are necessary to improve the results; (*ii*) the benefit of acquiring an edge in the first stage dissolves among the scenarios. The first explanation is corroborated by the average execution times exhibited in Table 4.2. The best results attained by BRKGA gets closer to the one hour limit as more

[2] Detailed results can be found in: http://www.ic.unicamp.br/~fusberti/sstp/

Table 2. BRKGA performance for instances with known optimums.

				BRKGA		
instance	k	$\lvert\overline{R}\rvert$	opt	cost	gap (%)	time
lin01_53_80	5	4.6	797	797	0	193.7
lin01_53_80	10	4.2	633.2	636	0.44	1027.4
lin01_53_80	20	4.6	753.9	754.4	0.07	2.2
lin01_53_80	50	4.7	768.9	769	0.01	1456.3
lin02_55_82	5	4.6	476.2	476.2	0	0.3
lin02_55_82	10	5.3	739.1	739.1	0	2529
lin02_55_82	20	5.3	752.2	752.2	0	4.6
lin02_55_82	50	5.1	732.6	732.8	0.03	803
lin03_57_84	5	4.4	653	655.2	0.34	0.3
lin03_57_84	10	5.2	834.7	840.6	0.71	38.4
lin03_57_84	20	5.8	854.9	855.5	0.07	583.4
lin03_57_84	50	5.5	895.7	896	0.03	502
lin04_157_266	5	10.4	1922.1	1923	0.05	44
lin04_157_266	10	9.8	1959.1	1972.4	0.68	1161.6
lin04_157_266	20	9.3	1954.9	1981.2	1.35	3125.3
lin04_157_266	50	9.8	2097.7	2172.5	3.57	2816.8
lin05_160_269	5	10.2	2215.5	2224.9	0.42	20.2
lin05_160_269	10	11.4	2210.2	2224.3	0.64	160.5
lin05_160_269	20	11.1	*2412.2**	2307	*-4.36**	1884.7
lin05_160_269	50	11.6	2297	2301.6	0.2	1645.4
lin06_165_274	5	11	1975.8	1986.9	0.56	411.9
lin06_165_274	10	10.6	1918.7	1935.1	0.85	389.4
lin06_165_274	20	14	2457.6	2463.9	0.26	702.9
lin06_165_274	50	12.6	2186.8	2196	0.42	3122.9

* refers to the best solution obtained by Bomze et al. [2010].
k – number of scenarios.
$\lvert\overline{R}\rvert$ – average number of terminals per scenario.
opt – optimum cost [2010].
cost – BRKGA best solution cost.
gap – optimality gap.
time – BRKGA execution time (sec) to obtain the best solution.

scenarios are introduced. This indicates that our results can be further improved with longer execution times. Figure 5 clearly shows this ascending/descending behaviors of execution time and cost decrease, respectively, with the inclusion of more scenarios.

Table 3. Summarized results for DIMACS benchmark.

	Instance Type							
	K		Lin		P		Wrp	
k	δc (%)	time	δc (%)	time	δc (%)	time	δc (%)	time
5	6.93	326.45	6.64	1354.54	5.50	896.61	3.34	963.04
10	5.90	408.86	4.88	900.92	4.81	448.41	2.86	1075.23
20	5.03	827.31	4.78	2046.42	4.55	1623.28	2.79	1689.66
50	3.90	2045.47	4.19	2836.25	3.62	1798.56	2.71	2198.19
75	4.02	2027.70	3.73	3368.88	3.70	2109.76	2.67	2657.79
100	3.97	2386.38	3.32	3116.50	3.73	2561.77	2.64	2828.89
150	3.77	2803.33	3.20	3317.81	3.91	3371.17	2.62	3274.75
200	3.80	3293.86	2.99	3293.12	3.69	3456.40	2.58	3341.13
250	3.89	3180.44	2.97	3390.52	3.68	3567.79	2.53	3452.78
300	3.73	3456.86	2.76	3440.17	3.68	3490.74	2.50	3531.54
400	3.52	3499.44	2.55	3532.98	3.61	3554.53	2.46	3545.27
500	3.56	3537.45	2.38	3541.23	3.49	3551.97	2.43	3571.08
750	3.41	3525.27	2.16	3583.80	3.06	3558.99	2.37	3504.00
1000	3.22	3559.16	1.96	3575.84	2.99	3567.02	2.34	3544.70

k – number of scenarios.
time – average execution time to achieve the best solution.
δc – average cost reduction compared to the *buy_none* solutions.

5 Final Remarks

This paper proposed a heuristic methodology for the stochastic Steiner tree problem that is capable of solving large instances within reasonable execution times. This task was achieved through a biased random-key genetic algorithm (BRKGA), for which we have designed efficient chromosome representations and solution evaluation techniques. Our implementation was able to solve instances up to 1000 scenarios and 613 edges, which represents a major increase on the sizes of the instances tackled in literature.

Computational experiments were performed with 24 small instances, previously solved by a state-of-the-art exact algorithm [2010]. The results have shown that the BRKGA achieved near optimal solutions, under one hour of computational time, giving an average gap of less than 0.5%. Additional experiments with the DIMACS benchmark, which complies 560 instances, confirmed the robustness of our approach. For these instances, the BRKGA achieved 3.42% of

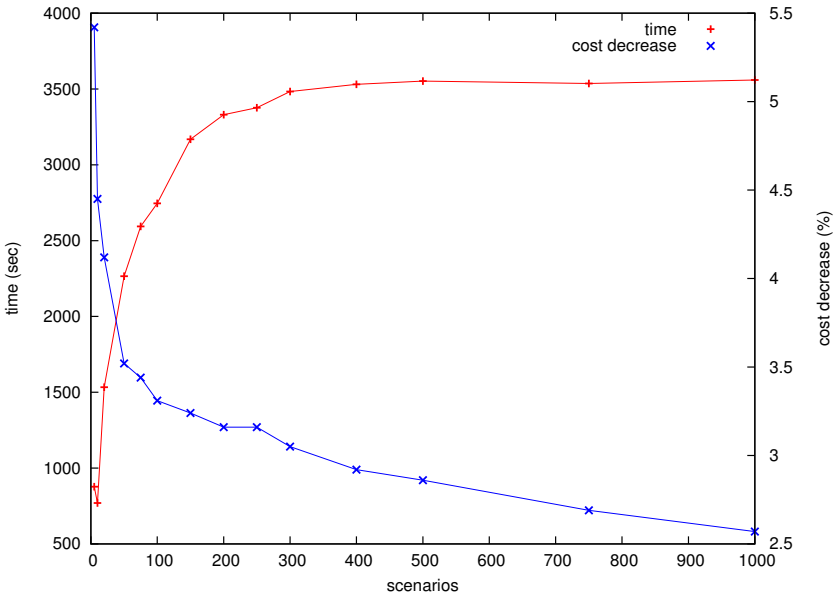

Fig. 5. The effects of the number of scenarios on the BRKGA results.

cost reduction from reference solutions for which no edge is bought in the first stage.

From the computational tests and the overall quality of the solutions obtained through our approach, it follows that the BRKGA is an effective methodology, capable of solving hard instances that may arise from real-life network design problems.

References

[2014] 11th DIMACS Implementation Challenge in Collaboration with ICERM: Steiner Tree Problems. http://dimacs11.cs.princeton.edu/. Last access in November 2014.

[1988] Birge, J.R., Louveaux, F.: A multicut algorithm for two-stage stochastic linear programs. European Journal of Operational Research 34, 384392 (1988)

[2010] Bomze, Immanuel M. and Chimani, Markus and Jnger, Michael and Ljubic, Ivana and Mutzel, Petra and Zey, Bernd.: Solving Two-Stage Stochastic Steiner Tree Problems by Two-Stage Branch-and-Cut. LNCS 6506, ISAAC (1), 427–439(2010)

[1967] Cavalli-Sforza, L. L. and Edwards, A.W.: Phylogenetic analysis. Models and estimation procedures. American Journal of Human Genetics, 19(3), 233–57 (1967)

[1989] Goldberg, David E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. Inc. 1st(1989)

[2011] Gonçalves, José Fernando and Resende, Mauricio G.: Biased Random-key Genetic Algorithms for Combinatorial Optimization. Journal of Heuristics, 17, 487–525 (2011)

[2007] Gupta, Anupam and Ravi, R. and Sinha, Amitabh: LP Rounding Approximation Algorithms for Stochastic Network Design. Math. Oper. Res. 32, 345–364 (2007).
[1993] Laporte, G. and Louveaux, F.: The integer L-shaped method for stochastic integer programs with complete recourse. per. Res. Lett. 13, 133142 (1993)
[2014] Lemon – Graph Library. http://lemon.cs.elte.hu/trac/lemon/. Last access in November 2014.
[1990] Lengauer, Thomas: Combinatorial Algorithms for Integrated Circuit Layout. John Wiley & Sons, Inc., New York (1990)
[1984] Magnanti, T. L. and Wong, R. T.: Network Design and Transportation Planning: Models and Algorithms. Transportation Science, 18(1), 1–55 (1984)
[2006] Swamy, Chaitanya and Shmoys, David B.: Approximation Algorithms for 2-stage Stochastic Optimization Problems. SIGACT News,37, 33–46 (2006)
[1969] Van Slyke R. M. and Roger Wets.: L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. SIAM Jornal on Applied Mathematics, Vol.17 No.4, 638–663 (1969)
[2003] V.V. Vazirani: Approximation Algorithms Springer
[1991] W. Spears, de K. Jong.: On the virtues of parameterized uniform crossover In Proceedings of the Fourth International Conference on Genetic Algorithms (1991), pp. 230–236