STRONG LOWER BOUNDS FOR THE PRIZE COLLECTING STEINER PROBLEM IN GRAPHS

ABILIO LUCENA AND MAURICIO G. C. RESENDE

ABSTRACT. Given an undirected graph G with nonnegative edges costs and nonnegative vertex penalties, the prize collecting Steiner problem in graphs (PCSPG) seeks a tree of G with minimum weight. The weight of a tree is the sum of its edge costs plus the sum of the penalties of those vertices not spanned by the tree. In this paper, we present an integer programming formulation of the PCSPG and describe an algorithm to obtain lower bounds for the problem. The algorithm is based on polyhedral cutting planes and is initiated with tests that attempt to reduce the size of the input graph. Computational experiments were carried out to evaluate the strength of the formulation through its linear programming relaxation. The algorithm found optimal solutions for 99 out of the 114 instances tested. On 96 instances, integer solutions were found (thus generating optimal PCSPG solutions). On all but seven instances, lower bounds were equal to best known upper bounds (thus proving optimality of the upper bounds). Of these seven instances, four lower bounds were off by 1 of the best known upper bound. Nine new best known upper bounds were produced for the test set.

1. INTRODUCTION

Let $G = (V, E)$ be an undirected graph with a set of vertices V and a set of edges E. Real-valued *costs* $\{c_e : e \in E\}$ are associated with the edges of G while real-valued *penalties* $\{d_v : v \in V\}$ are associated with the vertices of G. A tree is a connected acyclic subgraph of G and has a *weight* that equals the sum of its edge costs plus the sum of the penalties of those vertices of G that are not spanned by the tree. A solution of the prize collecting Steiner problem in graphs (PCSPG) is a minimum weight tree.

Applications for the PCSPG can be found, for example, in the design of local access telecommunication networks, where one wants to build a fiber-optic network for providing broadband connections to business and residential customers. The graph in this application corresponds to the local street map, with edges representing street segments and nodes representing street intersections and the locations of potential customer premises. The penalty associated with a node in this graph is an estimate of the potential loss of revenue that would result if that customer were not to receive service. Nodes corresponding to street intersections have penalties with zero value. The cost associated with an edge is the cost of laying the fiber on the corresponding street segment. Since labor and right-of-way costs greatly outweigh the cost of the fiber, one can assume that cable capacity is not a constraint.

Date: February 10, 2000. Revised October 11, 2002.
AT&T Labs Research Technical Report 00.3.1.

The PCSPG has been studied by Bienstock, Goemans, Simchi-Levi, and Williamson [3]. The roots for this problem can be traced to the prize collecting traveling salesman problem of Balas [2]. A 2-approximation algorithm for the PCSPG was proposed by Goemans and Williamson [9]. That improved on an earlier 5/2-approximation algorithm in [3]. An implementation of the Goemans and Williamson algorithm is given in Johnson, Minkoff, and Phillips [11], where extensive experimental results are provided. Canuto, Resende, and Ribeiro [4] present a multi-start heuristic that makes use of a randomized Goemans and Williamson algorithm with local search. Note that the PCSPG is related to the node weighted Steiner problem in graphs (NWSPG) (see Segev [19]). In fact, the NWSPG is a special case of PCSPG in that the single NWSPG terminal vertex can be conveniently treated as a PCSPG vertex with a sufficiently positive penalty. To the best of our knowledge, no exact solution algorithm specifically for PCSPG has been described in the literature.

In this paper, we introduce an integer programming formulation of the PCSPG. The formulation is used, in a cutting-plane algorithm, to obtain lower bounds for the problem. The formulation is described in Section 2. In Section 3, the cutting-plane algorithm is presented. Single-vertex solutions, which can be readily computed, are excluded from the formulation in Section 4. The preprocessing strategy is described in Section 5 and the implementation of orthogonal cuts is described in Section 6. Computational results for randomly generated PCSPG instances are reported in Section 7. A subset of these instances is generated to resemble a real-world application. In Section 8, concluding remarks are made.

2. Problem formulation

Given a set $S \subseteq V$ of vertices, let $E(S) \subseteq E$ be the set of edges with both endpoints in S. Associate a real-valued variable x_e with every edge $e \in E$ and denote by $x(E(S))$ the sum $\sum_{e \in E(S)} x_e$. In addition, associate a real-valued variable y_v with every vertex $v \in V$ and denote by $y(S),\ \ S \subseteq V$, the sum $\sum_{s \in S} y_s$. In order to introduce a formulation of the PCSPG, consider a polyhedral region $\mathcal{R}$, defined as

$$x(E) = y(V) - 1, \tag{1}$$

$$x(E(S)) \leq y(S \setminus \{s\}), \qquad s \in S, \ \ S \subseteq V, \tag{2}$$

$$0 \leq x_e \leq 1, \qquad e \in E,$$

$$0 \leq y_v \leq 1, \qquad v \in V.$$

An integer programming formulation of the PCSPG is given by

$$\min \left\{ \sum_{e \in E} c_e x_e + \sum_{v \in V} d_v (1 - y_v) : (x, y) \in \mathcal{R} \cap (\mathbb{R}^{|E|}, \mathbb{Z}^{|V|}) \right\}. \tag{3}$$

Formulation (3) follows from an extended formulation of the Steiner problem in graphs (SPG), introduced independently by Goemans [8], Lucena [13], and Margot, Prodon, and Liebling [16].

Constraint (1) imposes that the number of edges selected, $x(E)$, equals the number of edges required for a spanning tree of the implied subgraph, i.e. $y(V) - 1$. Constraints (2) are called *generalized subtour elimination constraints* (GSECs). They guarantee that the solution is cycle free. GSECs generalize subtour elimination constraints (SECs) (introduced by Dantzig, Fulkerson, and Johnson [5] for the traveling salesman problem). Notice that if $y_s = 1$, for all $s \in S$, then (2) reduces to a SEC. The set of feasible solutions for (3) corresponds to the set of all trees of G.

The above formulation can be seen as a generalization of the spanning tree polytope [7], by noting that if the vertices that appear in an optimal (minimum weight) tree are given, then the PCSPG reduces to finding a minimum spanning tree (MST) of the subgraph of G induced by those vertices.

3. Solving the linear programming relaxation

A linear programming (LP) relaxation for (3) is

$$\min \left\{ \sum_{e \in E} c_e x_e + \sum_{v \in V} d_v (1 - y_v) : (x, y) \in \mathcal{R} \right\}. \tag{4}$$

As there are exponentially many GSECs in (2), one may choose to exclude some or all of these inequalities from the linear program in an initial stage of the solution process. This is usually done by firstly defining a polyhedral region $\mathcal{R}_1 \supseteq \mathcal{R}$ and then optimizing over $\mathcal{R}_1$. An adequate choice of $\mathcal{R}_1$ is attained, for instance, by dropping all GSECs from the set of inequalities that define $\mathcal{R}$. The resulting polyhedral region is described as

$$\begin{aligned} &x(E) = y(V) - 1, \\ &0 \le x_e \le 1, \ e \in E, \\ &0 \le y_v \le 1, \ v \in V, \end{aligned}$$

and a valid LP lower bound for (3) (and for (4)) is

$$\min \left\{ \sum_{e \in E} c_e x_e + \sum_{v \in V} d_v (1 - y_v) : (x, y) \in \mathcal{R}_1 \right\}. \tag{5}$$

Let $(\bar{x}, \bar{y})$ be an optimal solution of (5). If $(\bar{x}, \bar{y})$ violates one or more GSECs, then these violated GSECs may be introduced as cutting planes. In the process, the following separation problem must be solved: Find a GSEC that is violated by $(\bar{x}, \bar{y})$ or determine that no such inequality exists. If no violated GSEC exists, then optimality of (4) is verified. Otherwise, the LP relaxation is reinforced with the introduction of violated GSECs and the corresponding LP is reoptimized. This is repeated until optimality of (4) is attained. Note that this procedure produces a sequence of nondecreasing valid lower bounds for (3).

The separation problem posed above requires the solution of at most $|V|$ maximum flow problems on a network with at most $|V|$ vertices. A procedure to solve it is described next.

3.1. **Separation of GSECs.** Let $(\bar{x}, \bar{y})$ denote the solution of the current LP relaxation. The *support graph* of this solution is the subgraph of G induced by vertices and edges with nonzero variables in $(\bar{x}, \bar{y})$. For a given vertex l in the support graph, let S_l be the subset S of V that contains l and maximizes $\bar{x}(E(S)) - \bar{y}(S) + \bar{y}_l$. Clearly, whenever this maximum is nonpositive, no violated inequality of type (2) that includes vertex l exists. Otherwise, S_l is associated with the most violated GSEC that contains vertex l. To determine S_l, the following quadratic Boolean problem must be solved:

$$\max \sum_{(u,v)\in E} \bar{x}_{(u,v)} z_u z_v - \sum_{v\in V} \bar{y}_v z_v + \bar{y}_l \tag{6}$$

subject to

$$z_l = 1, \tag{7}$$

$$z_v \in \{0,1\}, \qquad v \in V. \tag{8}$$

Variable z_v indicates whether vertex v is in set S ($z_v = 1$) or not ($z_v = 0$).

Problem (6)–(8) can be reformulated (cf. Picard and Ratliff [18]) as one of finding a maximum flow on a companion network of at most $|V| + 2$ vertices. It is thus solvable in polynomial time. An algorithm for the separation of SECs, due to Padberg and Wolsey [17], which is based on implicitly solving a problem similar to (6)–(8), can be easily adapted to separate GSECs [15]. This adaptation is reviewed below.

Let $N = (\bar{V} \cup \{s,t\}, A)$ be a companion network where $\bar{V}$ is the set of vertices in the support graph for $(\overline{x}, \overline{y})$. Nodes s and t are, respectively, a source and a sink node for N. Arc set A has, for every edge $e = (u,v)$ in the support graph, two arcs $[u,v]$ and $[v,u]$ of capacities $\xi_{uv} = \xi_{vu} = \bar{x}_e/2$. For all vertices v of the support graph, let $\delta(v)$ denote the set of edges incident to vertex v (in the support graph) and $\xi_v = \sum_{u\in\delta(v)} \xi_{vu}$. The source node has arcs $[s,v]$, for $v \in \bar{V}$, of capacity $\xi_{sv} = \max\{\xi_v - \bar{y}_v, 0\}$. The sink node, on the other hand, has arcs $[v,t]$, for $v \in \bar{V}$, of capacity $\xi_{v,t} = \max\{\bar{y}_v - \xi_v, 0\}$.

For a given vertex $l \in \bar{V}$, an optimal solution to (6)–(8) corresponds to a maximum flow (minimum cut) over $N = (\bar{V}\{s,t\}, A)$ with ξ_{sl} set to ∞. This problem can be solved to optimality in polynomial time [1]. Therefore, the separation for SECs is also solvable in polynomial time.

Following the procedure above, one may generate duplicate copies of violated subtours. To avoid this, one can (cf. [17]), instead, solve $|\overline{V}|$ maximum flow problems. For the k-th maximum flow problem, $\xi_{sk} = \infty$, and if $k \geq 2$, $\xi_{v,t} = \infty$ for $v = 1,\ldots,k-1$. In this way, if $|S| \geq 3$, then for all $k = 1,\ldots,|\overline{V}|$, $k \in S$ and $\{1,\ldots,k-1\} \in (\overline{V} \cup \{s,t\}) \setminus S$.

As many cuts as there are violated GSECs generated by the procedure above are introduced simultaneously for every very linear programming relaxation of (3) considered. In the computational results presented in Section 7, we call an *iteration* of the algorithm the process of solving one LP relaxation and solving the resulting separation problem.

4. Excluding single-vertex solutions

Instead of generating PCSPG lower bounds directly from the LP relaxation of (3), a different approach is followed. Notice that the most basic form of a PCSPG solution consists of a single, isolated, positive penalty vertex whose value can be computed efficiently. As a result, one may set aside single vertex solutions and restrict (3) to deal exclusively with solutions involving one or more edges (i.e. two or more vertices). That can be attained by considering a polyhedral region,

$\mathcal{R}_2$, defined by the inequalities that define $\mathcal{R}$ plus, for every $v \in V$, the following inequalities:

$$x(E(\delta(v))) \geq \begin{cases} y_v, & \text{if } d_v > 0, \\ 2y_v, & \text{if } d_v = 0. \end{cases} \tag{9}$$

Assume that at least one vertex $v \in V$ has a positive penalty and restrict $\mathcal{R}$ with inequalities (9). Under the objective function used and over the restricted solution space, PCSPG solutions are limited to one or more edges. That is indicated by the right hand side of the inequalities. For the case where $d_v > 0$, the inequality imposes that any positive penalty vertex in an optimal tree must be at least a leaf of that tree. If $d_v = 0$, then the inequality imposes that these vertices cannot be leaves (see Section 5 for the reasoning behind this restriction). Therefore, if any such vertex appears at an optimal tree, its (tree) edge degree must be greater than one.

An integer programming formulation for PCSPG (restricted to feasible solutions with one or more edges) is then given by

$$\min \left\{ \sum_{e \in E} c_e x_e + \sum_{i \in V} d_i (1 - y_i) : (x, y) \in \mathcal{R}_2 \cap (\mathbb{R}^{|E|}, \mathbb{Z}^{|V|}) \right\}. \tag{10}$$

We have found it computationally advantageous to split the set of feasible PCSPG solutions into the two subsets indicated above, namely single vertex solutions and multiple vertex solutions. This is because, in our computational experience, (10) appears to be a stronger formulation for the restricted PCSPG than (3) is for the unrestricted case. Even if this speculation proves incorrect, computing times to obtain LP relaxations for each formulation tend to be smaller for (10). As it may be appreciated in the section on computational results, in the majority of the instances tested, the solution of the LP relaxation of (10), i.e.

$$\min \left\{ \sum_{e \in E} c_e x_e + \sum_{i \in V \setminus T} d_i (1 - y_i) : (x, y) \in \mathcal{R}_2 \right\}, \tag{11}$$

turned out to be integral.

The lower bound given by (11) can be computed in a similar manner to that outlined in Section 3. Accordingly, we have chosen to optimize the objective function, at an initial stage, over a polyhedral region $\mathcal{R}_3$, defined as

$$\begin{aligned} x(E) &= y(V) - 1, \\ x(E(\delta(v))) &\geq y_v, v \in V, d_v > 0, \\ x(E(\delta(v))) &\geq 2y_v, v \in V, d_v = 0, \\ 0 &\leq x_e \leq 1, e \in E, \\ 0 &\leq y_v \leq 1,\ v \in V. \end{aligned}$$

The initial optimization is done with a primal simplex method. Every violated GSEC, obtained as detailed in Subsection 3.1, is appended to the set of inequalities defining $\mathcal{R}_3$ and the objective function is reoptimized over the resulting polyhedral region using the dual simplex method. Redefining $\mathcal{R}_3$ as the polyhedral region associated with the very last LP relaxation generated, the procedure i repeated until a stopping criterion is reached.

5. Reduction tests

A reduction test for the PCSPG attempts to determine vertices and edges that are guaranteed not to be in any optimal solution. Some simple reduction tests can be devised for the PCSPG. These tests follow directly from those that have been suggested for the SPG (see Duin [6], for instance). A description of the tests used to produce the results in Section 7 is given below.

5.1. **Shortest path test.** The shortest path test is only applied if $c_{uv} > 0$, for all $(u, v) \in E$. Let $\text{dist}(u, v)$ denote the length of the shortest path between vertices $u, v \in V$. If $\text{dist}(u, v) < c_{uv}$, then edge $(u, v) \in E$ can be eliminated from G.

5.2. **Cardinality-one test.** Assume that a given vertex, say vertex $v \in V$, has an edge cardinality of one, i.e. $|\delta(v)| = 1$. Denote the only edge incident to v by e. If $c_e > d_v$, then vertex v and, consequently, edge e cannot be in any optimal PCSPG tree. That applies because if edge e were in an optimal tree, its removal would lead to a feasible solution of a smaller cost, thus contradicting the optimality assumption. Indeed, this is the reason why a vertex $v \in V$ with $d_v = 0$ cannot be a leaf of an optimal tree (see inequalities (9)).

5.3. **Cardinality-two test.** Assume that a given vertex $v \in V$ has edge cardinality $|\delta(v)| = 2$, and denote the two edges incident to v by (v, v_1) and (v, v_2). If $d_v = 0$ and edges (v, v_1) and (v, v_2) have positive costs, either these two edges appear simultaneously in an optimal PCSPG solution or else neither can be in an optimal solution. The reasoning follows (as explained above) from the suboptimality of any PCSPG tree containing vertex v as a leaf.

Vertex v can be *pseudo eliminated* by being replaced by an edge (v_1, v_2) of cost $c_{v,v_1} + c_{v,v_2}$. In case multiple edges between two vertices result from this operation, only the one of least cost is kept.

5.4. **Cardinality-larger-than-two test.** The previous test can be extended to vertices $v \in V$ with $d_v = 0$ and $|\delta(v)| \geq 3$, where all edges incident to v have positive costs. For simplicity, assume that vertex v, under consideration, has $|\delta(v)| = 3$. The basic idea of this test is that if certain conditions are fulfilled, then vertex v is guaranteed to be either absent from an optimal tree or else to be in an optimal tree and have an edge degree equal to two.

For a vertex v as defined above, let v_1, v_2, and v_3 be the only three vertices that share edges with v. Accordingly, let e_1, e_2, and e_3 be the corresponding edges. If,

$$\begin{aligned}(12)\quad & \min\{\text{dist}(v_1, v_2) + \text{dist}(v_1, v_3), \text{dist}(v_2, v_1) + \text{dist}(v_2, v_3), \\ & \qquad \text{dist}(v_3, v_1) + \text{dist}(v_3, v_2)\} \leq c_{e_1} + c_{e_2} + c_{e_3},\end{aligned}$$

then vertex v is guaranteed not to have an edge degree of three in any optimal PCSPG tree (since a cheaper option of a lesser degree exists). Therefore, the edge degree of v in any optimal tree must be either 0 or 2 (see [6] for details). That applies since, as explained before, a PCSPG tree where v has an edge degree of one is suboptimal and the cost of the degree three solution (i.e. the rhs of (12)) is dominated by degree zero or two solutions (i.e. the lhs of (12)).

Whenever (12) is verified, one can pseudo eliminate vertex v. Once again, this is achieved by conveniently replacing every different combination of pairs of edges incident to v by an adequately chosen edge, as explained in the previous subsection.

This test can also be extended to vertices $v \in V$ with $|\delta(v)| > 3$ where all edges incident to v have positive costs. One should notice that the right hand side of (12) gives the MST cost for the subgraph of G induced by vertices v_1, v_2 and v_3 under edge costs given by the shortest path lengths in G: $\text{dist}(v_1, v_2)$, $\text{dist}(v_1, v_3)$ and $\text{dist}(v_2, v_3)$ (note that when $|\delta(v)| = 3$, such a spanning tree must have exactly two edges). The test would then amount to computing MSTs (under shortest path edge costs) for every possible combination (with three or more elements) of the vertices incident to v. Analogously to (12), MST costs should, in turn, be compared with the sum of the costs for the edges that link v directly to the MST vertices (in G). To be successful, MST costs should not exceed their sum of edge costs counterparts for any of the combinations tested.

Due to the combinatorial nature of test just described, in practice we limit it only to vertices with small edge cardinalities.

6. Orthogonal cuts

For a given LP relaxation of (3), it has been found computationally advantageous to adapt the separation algorithm of Section 3 to generate, at every iteration, cutting planes where vertices in a violated GSEC-defining subset do not appear in other violated GSEC-defining subsets generated at the current iteration. Cuts that do not have common variables are called *orthogonal cuts*. Introduction of orthogonal cuts have brought about substantial reductions in the CPU time required to compute the lower bounds. In some cases that amounted to reducing CPU times by a factor of 60.

Once again, let $(\bar{x}, \bar{y})$ denote the solution of the LP relaxation on hand. Orthogonal cuts are generated by first solving (6)-(8) for all vertices that are on the support graph of $(\bar{x}, \bar{y})$. Let S_l be the subset associated with the most violated GSEC found. The corresponding GSEC is then selected to be introduced as a cutting plane into $\mathcal{R}_3$ and all the vertices in S_l are eliminated from the support graph resulting in a restricted support graph. The procedure is recursively applied to the restricted support graph until no more violated GSECs are found. In spite of being clearly more computationally expensive than the previous separation scheme, it has shown to pay off by usually cutting overall CPU times. The cuts above are derived in the style of the *nested cuts* of Koch and Martin [12] for the SPG.

7. Computational Experiments

The main objective of this computational experiment was to evaluate the quality of the lower bounds generated by the cutting planes algorithm described in this paper. The algorithm was tested extensively on 114 test problems [1] described in [4, 11]. These problems range in size from 100 nodes and 284 edges to 1000 nodes and 25,000 edges. Tables 1–3 list these problems. For each instance, the tables list the instance name, the original dimension of the graph, the dimension of the graph after reduction with the procedures described in Section 5, and the best known upper bound prior to this paper (obtained by Canuto, Resende, and Ribeiro [4]).

The experiments were done on an SGI Challenge computer (28 196 MHz MIPS R10000 processors) with 7.6 Gb of memory. Each run used a single processor. The algorithm was coded in Fortran and uses primal and dual Simplex LP solvers in CPLEX 6.5 [10]. CPU times were measured with the system function `etime`.

[1] The test problems can be downloaded from `http://www.research.att.com/~mgcr/data`.

TABLE 1. Johnson, Minkoff, and Phillips [11] instances.

Problem	Original Nodes	Original Edges	Reduced Nodes	Reduced Edges	Upper Bound [4]
P100	100	317	86	212	803300
P100.1	100	284	91	211	926238
P100.2	100	297	83	201	401641
P100.3	100	316	94	243	659644
P100.4	100	284	83	221	827419
P200	200	587	172	447	1317874
P400	400	1200	361	1029	2459904
P400.1	400	1212	352	1025	2808440
P400.2	400	1196	364	1040	2518577
P400.3	400	1175	358	1008	2951725
P400.4	400	1144	356	972	2817438
K100	100	351	42	170	135511
K100.1	100	348	36	124	124108
K100.2	100	339	33	118	200262
K100.3	100	407	20	87	115953
K100.4	100	364	36	132	87498
K100.5	100	358	38	140	119078
K100.6	100	307	29	81	132886
K100.7	100	315	25	71	172457
K100.8	100	343	49	173	210869
K100.9	100	333	21	67	122917
K100.10	100	319	37	111	133567
K200	200	691	99	361	329211
K400	400	1515	237	944	350093
K400.1	400	1470	212	800	490771
K400.2	400	1527	217	935	477073
K400.3	400	1492	195	694	401881
K400.4	400	1426	190	747	389451
K400.5	400	1456	223	799	519526
K400.6	400	1576	239	986	374849
K400.7	400	1442	225	883	474466
K400.8	400	1516	245	1036	418614
K400.9	400	1500	205	803	383105
K400.10	400	1507	211	855	394191

The code was compiled with the SGI MIPSpro F77 compiler using flags `-Ofast -static`.

Tables 4–6 summarize the computational results. Note that all instances tested have integral valued edge costs and vertex penalties. Therefore, fractional LP relaxation lower bounds can be rounded up to integrality. For each instance, the table lists the instance name, the best known upper bound, with new best known bounds found by the cutting planes algorithm indicated in bold, the least number of iterations (i.e. number of GSEC separation rounds) and CPU times to reach best (integral valued) PCSPG lower bounds (rounding up fractional LP relaxation values, if necessary), the number of iterations to reach the optimal solution, the objective function value of the optimal LP relaxation, the corresponding integral valued PCSPG lower bound, and an indication as to whether or not the best LP relaxation solution is integral. Instances for which the cutting planes method was not able to find an optimal solution are indicated by the "did not finish" label.

TABLE 2. Steiner series C test instances.

Problem	Original Nodes	Original Edges	Reduced Nodes	Reduced Edges	Upper Bound [4]
C1-A	500	625	116	214	18
C1-B	500	625	125	226	85
C2-A	500	625	110	209	50
C2-B	500	625	112	211	141
C3-A	500	625	174	293	414
C3-B	500	625	204	325	737
C4-A	500	625	207	331	618
C4-B	500	625	247	371	1063
C5-A	500	625	254	375	1080
C5-B	500	625	326	447	1528
C6-A	500	1000	356	823	18
C6-B	500	1000	356	823	55
C7-A	500	1000	366	843	50
C7-B	500	1000	366	843	103
C8-A	500	1000	382	866	361
C8-B	500	1000	385	869	500
C9-A	500	1000	412	903	533
C9-B	500	1000	416	907	694
C10-A	500	1000	431	920	859
C10-B	500	1000	440	929	1069
C11-A	500	2500	489	2143	18
C11-B	500	2500	489	2143	32
C12-A	500	2500	485	2189	38
C12-B	500	2500	485	2189	46
C13-A	500	2500	488	2167	237
C13-B	500	2500	488	2167	258
C14-A	500	2500	493	2168	293
C14-B	500	2500	493	2168	318
C15-A	500	2500	496	2153	501
C15-B	500	2500	496	2153	551
C16-A	500	12500	500	4740	11
C16-B	500	12500	500	4740	11
C17-A	500	12500	500	4704	18
C17-B	500	12500	500	4704	18
C18-A	500	12500	500	4781	111
C18-B	500	12500	500	4781	113
C19-A	500	12500	500	4729	146
C19-B	500	12500	500	4729	146
C20-A	500	12500	500	4770	266
C20-B	500	12500	500	4770	267

The optimal solution of the LP relaxation was found for 99 of the 114 test instances. On 96 of those 99 instances, the optimal solutions were integral, thus solving the PCSPG. On all but seven of the 114 instances, lower bounds were equal to known upper bounds, thus proving optimality of the upper bounds. Of these seven instances, four had lower bounds that were off of the upper bounds by a unit. The relative error of the remaining three instances with respect to the best known upper bound was never greater than 1.4%. Nine new best known upper bounds were found with the cutting planes algorithm.

In the remainder of this section, we make further comments about the computational experiments, considering each problem class individually.

Table 3. Steiner series D test instances.

	Original		Reduced		Upper
Problem	Nodes	Edges	Nodes	Edges	Bound [4]
D1-A	1000	1250	233	443	18
D1-B	1000	1250	233	443	106
D2-A	1000	1250	261	485	50
D2-B	1000	1250	264	488	228
D3-A	1000	1250	340	571	807
D3-B	1000	1250	400	634	1510
D4-A	1000	1250	381	616	1203
D4-B	1000	1250	458	694	1881
D5-A	1000	1250	521	768	2157
D5-B	1000	1250	660	907	3135
D6-A	1000	2000	741	1709	18
D6-B	1000	2000	741	1709	70
D7-A	1000	2000	735	1706	50
D7-B	1000	2000	736	1707	105
D8-A	1000	2000	794	1772	755
D8-B	1000	2000	800	1780	1038
D9-A	1000	2000	791	1758	1072
D9-B	1000	2000	800	1767	1420
D10-A	1000	2000	844	1825	1671
D10-B	1000	2000	860	1842	2079
D11-A	1000	5000	986	4658	18
D11-B	1000	5000	986	4658	30
D12-A	1000	5000	992	4641	42
D12-B	1000	5000	992	4641	42
D13-A	1000	5000	990	4614	445
D13-B	1000	5000	990	4614	486
D14-A	1000	5000	991	4621	602
D14-B	1000	5000	991	4621	665
D15-A	1000	5000	993	4622	1042
D15-B	1000	5000	993	4622	1108
D16-A	1000	25000	1000	10595	13
D16-B	1000	25000	1000	10595	13
D17-A	1000	25000	1000	10542	23
D17-B	1000	25000	1000	10542	23
D18-A	1000	25000	1000	10312	218
D18-B	1000	25000	1000	10312	224
D19-A	1000	25000	1000	10242	308
D19-B	1000	25000	1000	10242	311
D20-A	1000	25000	1000	10471	536
D20-B	1000	25000	1000	10471	537

7.1. **Johnson, Minkoff, and Phillips test problems.** We consider 34 test problems introduced by Johnson, Minkoff, and Phillips [11]. Reduction tests have a significant but decreasing impact as instance dimension grows. This can be observed in Table 1 for the Johnson, Minkoff, and Phillips test problems. On all instances, the algorithm found optimal solutions. All optimal solutions were integral, thus producing feasible upper bounds. In five of the 34 instances, a single vertex solution having a better objective function value than the optimal solution of the relaxation was found.

7.2. **Steiner series C test problems.** We considered 40 instances of the Steiner series C test problems, proposed in Canuto, Resende, and Ribeiro [4]. Reduction

Table 4. Computational results: Johnson, Minkoff, and Phillips [11] instances.

Problem	Upper Bound	To bound Iter.	To bound Time (s)	To optimal Iter.	To optimal Time (s)	Lower Bound	Solution Integral?
P100	803300	38	0.51	38	0.51	803300	yes
P100.1	926238	41	0.53	41	0.53	926238	yes
P100.2	401641	46	0.37	46	0.37	401641	yes
P100.3	659644	56	0.49	56	0.49	659644	yes
P100.4	827419	44	0.32	44	0.32	827419	yes
P200	1317874	58	1.47	59	1.56	1317874	yes
P400	2459904	972	326.85	972	326.85	2459904	yes
P400.1	2808440	812	294.19	812	294.19	2808440	yes
P400.2	2518577	387	68.57	387	68.57	2518577	yes
P400.3	2951725	373	52.56	373	52.56	2951725	yes
P400.4	2817438	393	82.49	393	82.49	2817438	yes
K100	135511	Single vertex		131	0.88	135511	yes
K100.1	124108	Single vertex		98	0.66	124108	yes
K100.2	200262	131	1.21	132	1.22	200262	yes
K100.3	115953	93	0.56	93	0.56	115953	yes
K100.4	87498	Single vertex		25	0.15	87498	yes
K100.5	119078	85	0.50	86	0.51	119078	yes
K100.6	132886	39	0.19	39	0.19	132886	yes
K100.7	172457	59	0.30	59	0.30	172457	yes
K100.8	210869	163	1.53	163	1.53	210869	yes
K100.9	122917	Single vertex		32	0.15	122917	yes
K100.10	133567	Single vertex		39	0.22	133567	yes
K200	329211	560	36.58	560	36.58	329211	yes
K400	350093	1687	905.61	1687	905.61	350093	yes
K400.1	490771	1967	5276.93	1967	846.44	490771	yes
K400.2	477073	1897	810.05	1897	810.05	477073	yes
K400.3	401881	1111	322.61	1111	322.61	401881	yes
K400.4	389451	1187	307.15	1188	308.21	389451	yes
K400.5	519526	1612	694.11	1612	694.11	519526	yes
K400.6	374849	1650	936.12	1650	936.12	374849	yes
K400.7	474466	2125	965.05	2125	965.05	474466	yes
K400.8	418614	1402	786.76	1402	786.76	418614	yes
K400.9	383105	1376	379.30	1376	379.30	383105	yes
K400.10	394191	2562	1081.74	2566	1083.54	394191	yes

tests have a significant but decreasing impact as instance dimension grows. This can be observed in Tables 2 for the Steiner series C test problems. For 4 instances in this set, the run was terminated prior to convergence to an optimal solution (see Table 7 for details on the runs that were terminated prior to convergence). Out of the 36 instances where optimal solutions were found, integral valued LP relaxations were computed for all but one instance (C18-A), thus producing optimal solutions to the PCSPG. For instance C18-A, where a fractional LP relaxation value was obtained, as well as in three of the four instances terminated prior to convergence to an optimal solution, rounding up the final objective function value matches a known upper bound for the instance (thus proving optimality of the upper bound). In the remaining instance (C20-A), the lower bound found was off by a unit from the best known upper bound. The cutting planes algorithm produced new best known upper bounds for two instances (C7-B and C13-A). In five of the 40 instances, a

Table 5. Computational results: Steiner series C test instances.

Problem	Upper Bound	To bound Iter.	To bound Time (s)	To optimal Iter.	To optimal Time (s)	Lower Bound	Solution Integral?
C1-A	18	Single vertex		8	0.12	18	yes
C1-B	85	171	1.50	186	1.72	85	yes
C2-A	50	Single vertex		6	0.12	50	yes
C2-B	141	82	0.92	86	0.96	141	yes
C3-A	414	64	0.98	80	1.18	414	yes
C3-B	737	105	19.68	133	26.12	737	yes
C4-A	618	68	1.50	73	1.68	618	yes
C4-B	1063	436	201.77	536	287.42	1063	yes
C5-A	1080	223	47.71	314	80.36	1080	yes
C5-B	1528	1568	2612.08	1966	3487.08	1528	yes
C6-A	18	Single vertex		19	0.94	18	yes
C6-B	55	829	41.92	907	57.45	55	yes
C7-A	50	Single vertex		11	1.30	50	yes
C7-B	**102**	153	4.62	155	4.74	102	yes
C8-A	361	352	27.97	399	33.37	361	yes
C8-B	500	384	98.61	594	215.01	500	yes
C9-A	533	257	39.59	441	84.06	533	yes
C9-B	694	968	595.95	1911	1912.56	694	yes
C10-A	859	224	71.06	340	160.28	859	yes
C10-B	1069	821	1302.84	1674	3502.29	1069	yes
C11-A	18	Single vertex		52	3.44	18	yes
C11-B	32	413	25.13	776	68.53	32	yes
C12-A	38	386	21.18	596	37.03	38	yes
C12-B	46	721	109.73	815	126.66	46	yes
C13-A	**236**	529	235.0	858	332.52	236	yes
C13-B	258	1132	257.07	2906	3092.71	258	yes
C14-A	293	894	346.28	2195	1749.75	293	yes
C14-B	318	1079	1139.95	1710	1142.27	318	yes
C15-A	501	4524	7379.22	17283	54223.30	501	yes
C15-B	551	8213	23719.36	did not finish		551	?
C16-A	11	665	89.16	1244	204.50	11	yes
C16-B	11	665	89.38	1244	205.05	11	yes
C17-A	18	428	64.41	745	250.30	18	yes
C17-B	18	830	361.60	853	388.25	18	yes
C18-A	111	927	822.68	10850	20031.81	111	no
C18-B	113	1284	1526.48	did not finish		113	?
C19-A	146	1099	683.23	50067	152217.06	146	yes
C19-B	146	330	173.18	13041	18999.60	146	yes
C20-A	266	22187	165521.17	did not finish		265	?
C20-B	267	71476	932619.06	did not finish		267	?

single vertex solution having a better objective function value than the optimal solution of the relaxation was found.

7.3. **Steiner series D test problems.** We considered 40 instances of the Steiner series D test problems, proposed in Canuto, Resende, and Ribeiro [4]. Reduction tests have a significant but decreasing impact as instance dimension grows. This can be observed in Tables 3 for the Steiner series D test problems. For 11 instances in this set, the run was terminated prior to convergence to an optimal solution (see Table 7 for details on the runs that were terminated prior to convergence). Out of the 29 instances where optimal solutions were found, integral valued LP relaxations

Table 6. Computational results: Steiner series D test instances.

	Upper	To bound		To optimal		Lower	Solution
Problem	Bound	Iter.	Time (s)	Iter.	Time (s)	Bound	Integral?
D1-A	18	Single vertex		17	0.43	18	yes
D1-B	106	253	4.95	269	5.53	106	yes
D2-A	50	Single vertex		14	0.65	50	yes
D2-B	**218**	68	1.72	95	2.24	218	yes
D3-A	807	195	11.56	197	12.22	807	yes
D3-B	**1509**	237	206.49	342	331.50	1509	yes
D4-A	1203	236	34.64	281	51.50	1203	yes
D4-B	1881	533	1245.67	653	1551.29	1881	yes
D5-A	2157	349	290.06	599	597.47	2157	yes
D5-B	3135	9741	184501.31	did not finish		3135	?
D6-A	18	Single vertex		15	2.62	18	yes
D6-B	**67**	1420	155.65	1549	225.75	67	yes
D7-A	50	Single vertex		17	4.31	50	yes
D7-B	**103**	585	50.71	989	154.11	103	yes
D8-A	755	342	110.28	492	170.69	755	yes
D8-B	**1036**	1092	1238.95	2037	3267.93	1036	yes
D9-A	**1070**	1073	911.78	1321	1346.68	1070	no
D9-B	1420	3402	13241.68	5140	25052.53	1420	yes
D10-A	1671	2182	8526.87	7476	62590.02	1671	yes
D10-B	2079	7365	107820.08	did not finish		2079	?
D11-A	**18**	Single vertex		463	33.88	18	yes
D11-B	29	1410	419.83	1759	870.60	29	yes
D12-A	42	917	161.54	1092	281.74	42	yes
D12-B	42	913	161.62	1095	297.14	42	yes
D13-A	445	1395	1564.01	7595	24689.71	445	yes
D13-B	486	1000	1063.67	2242	4464.19	486	yes
D14-A	602	10584	47907.67	did not finish		602	?
D14-B	665	16340	104869.80	did not finish		665	?
D15-A	1042	8418	95251.23	did not finish		1040	?
D15-B	1108	25464	731657.0	38461	1691918.5	1107	no
D16-A	13	1222	272.46	2999	9957.77	13	yes
D16-B	13	1221	273.68	2540	6129.69	13	yes
D17-A	23	1492	52541.44	3609	16939.77	23	yes
D17-B	23	1318	116752.22	3171	13742.37	23	yes
D18-A	218	11997	79384.02	did not finish		218	?
D18-B	224	1547	4508.89	did not finish		223	?
D19-A	308	2892	12483.63	did not finish		306	?
D19-B	311	5519	29378.93	did not finish		310	?
D20-A	536	6	171.96	did not finish		529	?
D20-B	537	5	166.31	did not finish		530	?

were computed for all but two instances (D9-A and D15-B), thus producing optimal solutions to the PCSPG. For instance D9-A, where a fractional LP relaxation value was obtained, as well as in five of the 11 instances terminated prior to convergence to an optimal solution, rounding up the final objective function value matches a known upper bound for the instance (thus proving optimality of the upper bound). In the remaining instance where a fractional LP relaxation value was obtained (D15-B), as well as two instances terminated prior to convergence to an optimal solution (D18-B and D19-B), the lower bound found was off by a unit from the best known upper bound. The cutting planes algorithm produced new best known upper bounds for seven instances (D2-B, D3-B, D6-B, D7-B, D8-B, D9-A, and D11-A). In

Table 7. Computational results: Instances which did not terminate with optimal solution of LP relaxation. For each instance, the table lists number of iterations performed, total CPU time (in seconds) expended, objective function value at termination, the best known upper bound, and the relative error of the objective function value with respect to the best known upper bound.

Problem	Iterations	Time (s)	Solution	Upper Bound	Relative Error (%)
C15-B	12366	46280.32	550.163	551	0.152
C18-B	6450	65002.18	112.204	113	0.704
C20-A	68366	660808.88	264.927	266	0.403
C20-B	72335	952436.88	266.011	267	0.370
D5-B	10248	201439.42	3134.120	3135	0.028
D10-B	14946	463579.75	2078.193	2079	0.039
D14-A	51117	661451.50	601.993	602	0.001
D14-B	41898	534742.13	664.796	665	0.031
D15-A	9685	121711.20	1039.204	1042	0.268
D18-A	37088	423244.63	217.161	218	0.385
D18-B	95603	1955244.75	222.919	224	0.483
D19-A	58235	1218375.75	305.995	308	0.651
D19-B	42012	667919.38	309.56	311	0.463
D20-A	4150	172934.42	528.997	536	1.307
D20-B	4034	171829.67	529.997	537	1.304

five of the 40 instances, a single vertex solution having a better objective function value than the optimal solution of the relaxation was found.

8. Concluding Remarks

A formulation of the prize collecting Steiner problem in graphs was introduced in this paper. The strength of this formulation, as measured in terms of the lower bounds obtained from its LP relaxation, was tested on 114 randomly generated instances. The algorithm found optimal solutions for 99 out of the 114 instances tested. On 96 instances, integer solutions were found (thus generating optimal PCSPG solutions). On all but seven instances, lower bounds were equal to best known upper bounds (thus proving optimality of the upper bounds). Of these seven instances, four lower bounds were off by a unit of the best known upper bound. Since the upper bounds for these instances were produced heuristically [4], it is conceivable that these upper bounds are not optimal. Recall that nine new best known upper bounds were produced for the test set, improving upon the bounds in [4]. Consequently, one or more of these seven lower bounds may turn out to be optimal.

The instances of the Steiner Problem in Graph (SPG) from which PCSPG instances C and D are derived are today recognized as being easy to solve. Our computational experience with a lower bounding LP relaxation SPG algorithm which uses GSECs [14] indicates that these SPG instances are much easier to solve than the corresponding PCSPG instances. Clearly, for many of the instances tested, optimality could have been proven a lot earlier if known PCSPG upper bounds from the literature were used in conjunction with the lower bounds being generated. Nevertheless, even if this action were to be taken, the remark above would remain valid.

References

[1] N.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[3] D. Bienstock, M.X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59:413–420, 1993.

[4] S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbations for the prize collecting Steiner tree problem in graphs. *Networks*, 38:50–58, 2001.

[5] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson. Solution of a large scale traveling salesman problem. *Operations Research*, 2:393–410, 1954.

[6] C. Duin. *Steiner's problem in graphs*. PhD thesis, University of Amsterdam, 1994.

[7] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971.

[8] M.X. Goemans. The Steiner tree polytope and related polyhedra. *Mathematical Programming*, 63:157–182, 1994.

[9] M.X. Goemans and D.P. Williamson. The primal dual method for approximation algorithms and its application to network design problems. In D.S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 144–191. P.W.S. Publishing Co., 1996.

[10] ILOG CPLEX, Inc. Cplex 6.5, 1999.

[11] D.S. Johnson, M. Minkoff, and S. Phillips. The prize collecting tree problem: Theory and practice. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 760–769, 1999.

[12] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.

[13] A. Lucena. Tight bounds for the Steiner problem in graphs, 1991. Talk given at the TIMS XXX - SOBRAPO XXIII Joint International Meeting, Rio de Janeiro.

[14] A. Lucena. Tight bounds for the Steiner problem in graphs. Technical report, IRC for Process Systems Engineering, Imperial College, 1993.

[15] A. Lucena and J.E. Beasley. Branch and cut algorithms. In J.E. Beasley, editor, *Advances in linear and integer programming*, pages 187–221. Oxford University Press, 1996.

[16] F. Margot, A. Prodon, and Th.M. Liebling. Tree polyhedron on 2-tree. *Mathematical Programming*, 63:183–192, 1994.

[17] M. Padberg and L. Wolsey. Trees and cuts. *Annals of Discrete Mathematics*, 17:511–517, 1983.

[18] J.C. Picard and H.D. Ratliff. A graph theoretic equivalence for integer programs. *Operations Research*, 2:261–269, 1973.

[19] A. Segev. The node-weighted Steiner tree problem. *Networks*, 17:1–17, 1987.

(Abilio Lucena) Laboratório de Métodos Quantitativos, Departamento de Administração, Universidade Federal do Rio de Janeiro, Av. Pasteur 250, 22290-240 Rio de Janeiro, RJ, Brazil

E-mail address: lucena@openlink.com.br

(Mauricio G. C. Resende) Internet and Network Systems Research, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.

E-mail address: mgcr@research.att.com