New Benchmark Instances for the Steiner Problem in Graphs

Isabel Rosseti
Marcus Poggi de Aragão
Celso C. Ribeiro
Eduardo Uchoa
Renato F. Werneck
Department of Computer Science, Catholic University of Rio de Janeiro
Rua Marquês de São Vicente, 225, Rio de Janeiro, 22453-900, Brazil.
{rosseti, poggi, celso, uchoa, rwerneck}@inf.puc-rio.br

August 24, 2001

Abstract. We propose in this work 50 new test instances for the Steiner problem in graphs. These instances are characterized by large integrality gaps (between the optimal integer solution and that of the linear programming relaxation) and symmetry aspects which make them harder to both exact methods and heuristics than the test instances currently in use for the evaluation and comparison of existing and newly developed algorithms. Our computational results indicate that these new instances are not amenable to reductions by current preprocessing techniques and that not only do the linear programming lower bounds show large gaps, but they are also hard to be computed. State-of-the-art heuristics, which found optimal solutions for almost all test instances currently in use, faced much more difficulties for the new instances. Fewer optimal solutions were found and the numerical results are more discriminant, allowing a better assessment of the effectiveness and the relative behavior of different heuristics.

Keywords: Steiner problem in graphs, benchmark instances, test instances, algorithms

1. Introduction

Let $G = (V,E)$ be a connected undirected graph, where V is the set of nodes and E denotes the set of edges. Given a non-negative weight function $w : E \rightarrow \mathbb{R}_+$ associated with its edges and a subset $X \subseteq V$ of terminal nodes, the Steiner problem in graphs (SPG) consists of finding a minimum weighted subtree of G spanning all nodes in X. The solution of SPG is a Steiner minimum tree. This is one of the most widely studied NP-hard problems, with many applications.

Three sets of benchmark instances are commonly used in the literature to assess the performance of algorithms for the SPG: instances available from the online repository OR-Library (Beasley, 1990), the "incidence" instances of Duin (1994) (see also (Duin and Voss, 1997)), and the VLSI instances of Koch and Martin (1998). All these instances are available from the SteinLib repository (Koch et al., 2001). However, these instances are rapidly becoming inadequate for benchmarking, for the following reasons:

- They have already been solved to optimality, with the exception of a few of the larger incidence instances (Duin, 1994; Koch and Martin, 1998; Lucena and Beasley, 1998; Uchoa et al., 1999; Polzin and Vahdati, 2000; Poggi et al., 2001b). In fact, optimality can be proven within seconds in most cases. These instances are not challenging enough to stimulate further development of exact algorithms.

- Metaheuristics can easily find optimal solutions to a large portion of these instances (Duin and Voss, 1997; Duin and Voss, 1999; Gendreau et al., 1999; Martins et al., 2000; Ribeiro and Souza, 2000; Bastos and Ribeiro, 2001; Ribeiro et al., 2001). It is becoming increasingly hard to compare different metaheuristics properly. Deciding among different variants, parameter values, and implementation choices is also difficult, since they often lead to quite similar solutions, which are frequently optimal.

This paper proposes three new series of benchmark instances that will hopefully lead to a better assessment of exact and approximate algorithms for the SPG. Even though some of them are somewhat artificial, we believe they can play an important role in the development of algorithms whose ultimate goal is solving real-world instances. Algorithms that can cope with a variety of artificially hard instances tend to be very robust.

Our first goal was to design instances hard to be solved exactly. Current state-of-the-art exact algorithms are based on linear programming formulations that yield very tight bounds for most existing benchmark instances. They can solve several of these instances to optimality without branching. Even when branching is necessary, it is often possible to reduce the problem size significantly using reduction tests and fixation by reduced costs. Therefore, we tried to create instances with large duality gaps. To make them even harder, we introduced various degrees of symmetry, both in the structure of the underlying graphs and in terminal placement. In practice, symmetry increases the number of nodes in the search trees of branch-and-bound algorithms, since LP-based lower bounds tend to improve very slowly as branchings are performed.

The other design goal was to make the instances challenging also for metaheuristics. For each instance originally created, we generated another one with the same structure, but with perturbed edge weights. Since this results in instances with a much smaller number of optimal solutions, finding one of them by usual heuristic search methods tends to be a harder task.

Three classes of instances with a total of 50 reasonably small test problems are proposed and described in Section 2. The number of nodes in each graph ranges from 64 to 4096, while the number of edges ranges from 128 to 28512. This means that, although much harder, our instances are not bigger than those currently in use. Section 3 presents some computational results and discusses the effectiveness of the new instances in terms of achieving their goal. Concluding remarks are made in Section 4.

2. Instances

2.1. HYPERCUBE (hc)

Graphs in this series are d-dimensional hypercubes, with $d \in \{6,\dots,12\}$. For each value of d, the corresponding graph has 2^d nodes and $d \cdot 2^{d-1}$ edges. These graphs are bipartite (because all cycles are even, see e.g. (West, 2000)) and both partitions have the same number of nodes. The vertices in one of such subsets become terminals ($|X| = 2^{d-1}$). Edge weights in the originally created instances are unitary. The perturbed instances have integer edge weights randomly chosen from a uniform distribution in the interval $[100,110]$. These instances seem to be extremely difficult for existing algorithms. For example, using the branch-and-ascent algorithm proposed by Poggi et al. (2001b), we could not solve to optimality the instances with more than 128 nodes. Duality gaps are large and symmetry makes traditional branching schemes much less effective.

The naming convention is hc$d$[u|p], where u stands for "unperturbed" and p for "perturbed". For example, hc8u corresponds to an 8-dimensional hypercube with unperturbed weights. The pseudocode in Figure 1 describes the algorithm applied for the construction of these instances. We denote by $a \otimes b$ the integer number obtained by the exclusive-or operation between the binary representations of the integers a and b. To make the description of the algorithm simpler, the nodes of the d-dimensional hypercube are indexed by $i = 0,1,\dots,2^{d-1}-1$, instead of by $i = 1,2,\dots,2^{d-1}$. Then, $i \otimes 2^k$ gives the index of each neighbor of node i, for $k = 0,\dots,d-1$. Figure 2 shows what hc3u (a three-dimensional hypercubic instance) would look like.

2.2. CODE COVERING (cc)

Let V_q^n be the set of all n-dimensional vectors whose components are integers in the interval $[0,q-1]$. The *Code Covering Problem* (CCP) is defined as follows: given V_q^n and a positive integer r, find a minimum cardinality subset C of V_q^n such that there exists a vector $x \in C$ with $d(x,y) \leq r$ for all $y \in V_q^n$ (where d denotes the Hamming distance). This NP-hard problem (Lint, 1975) is equivalent to finding a minimum dominating set in a graph in which there is a vertex associated to each element of V_q^n and an edge connecting the nodes associated to every pair $x,y \in V_q^n$ such that $d(x,y) \leq r$.

We created instances for the SPG using 13 of such graphs and taking as the set of terminals a solution (i.e., a dominating set) obtained by the tabu search algorithm of Poggi and Souza (1999). Edges have incidence weights

$E \leftarrow \emptyset$
$V \leftarrow \emptyset$
$X \leftarrow \{0\}$
for $i = 0, \ldots, 2^d - 1$ do
    $V \leftarrow V \cup \{i\}$
    for $k = 0, \ldots, d - 1$ do
        $j \leftarrow i \otimes 2^k$
        if $i \notin X$ then $X \leftarrow X \cup \{j\}$
        $w_{i,j} \leftarrow 1$
        if $(i, j) \notin E$ then $E \leftarrow E \cup \{(i, j)\}$
    end_for
end_for

Figure 1. Algorithm for the generation of the hypercube instances.

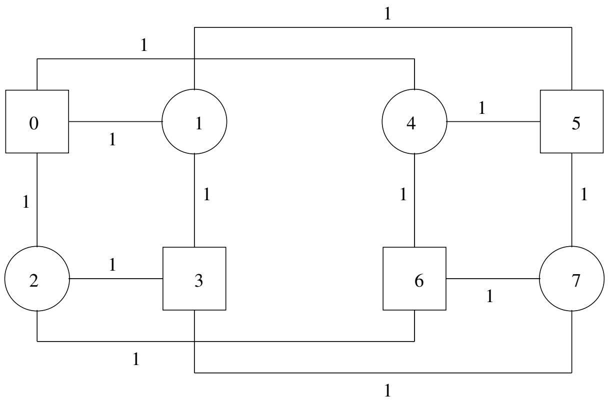

Figure 2. Graph associated with the hypercube instance hc3u.

so as to make preprocessing ineffective, following Duin (1994) and Duin and Voss (1997). In the unperturbed case, this means that an edge has weight equal to 1 if it connects non-terminals, 2 if it is incident to a single terminal, and 3 if it connects two terminals. The perturbed case follows the same principle, but with integral weights uniformly distributed in the intervals $[100, 110]$, $[200, 210]$, or $[300, 310]$, depending on the number of terminals incident to an edge.

Although these instances do challenge current exact algorithms, they are not as hard as the hc instances. The graphs in this case are still very symmetric, but terminal placement is not.

The naming convention is cc$n$-q[u|p] (the value of r is omitted because we have used only $r = 1$). For example, cc3-4u corresponds to an unperturbed

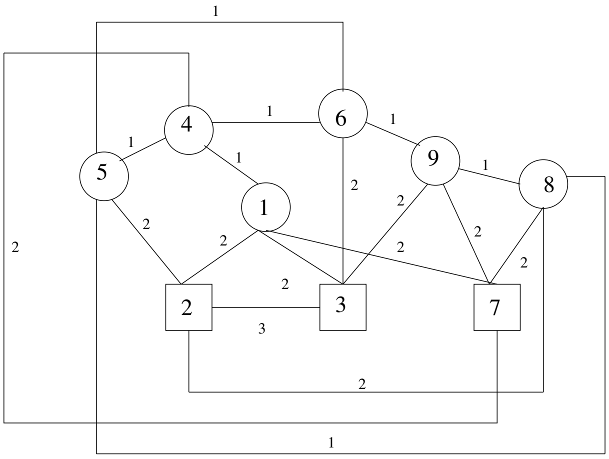

Figure 3. Code covering instance cc3-2u.

instance derived from a CCP with $n = 3$ and $q = 4$ (and $r = 1$). Figure 3 shows what the code covering instance cc3-2u with $q = 2$ and $n = 3$ would look like.

2.3. BIPARTITE (bip)

This series contains ten instances defined on irregular bipartite graphs, with one of the bipartition-defining vertex subsets acting as the set of terminals. These graphs are derived from random instances of the set covering problem (SCP) from the OR-Library (Beasley, 1990). Given an instance I of SCP with m rows and n columns, we build an instance I' for the SPG as follows. There is a terminal in I' associated with each row i in I. Similarly, each non-terminal node in I' is associated with a column j in I. Whenever row i covers column j, there is an edge linking terminal i to the non-terminal node j in I'. The resulting graph has $m+n$ vertices, m terminals, and as many edges as nonzero entries in the coefficient matrix of the SCP instance I. Edge weights are unitary in the unperturbed case and uniformly distributed in the interval $[100, 110]$ for the instances with perturbations.

The naming convention is bip$I$[u|p], where I is the name of the SCP instance (minus the scp prefix). For example, bipe2u is the SPG instance with unperturbed weights associated with instance scpe2 of the OR-Library. Figure 4 shows what a bipartite instance with $m = 3$ and $n = 4$ would look like.

Our motivation in creating this series was to have instances with a certain structure (bipartite), but without the artificial symmetry found in the previous two classes. Using SCP instances is no better or worse than generating ran-

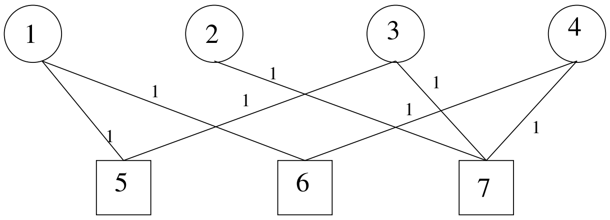

Figure 4. An example of bipartite instance.

dom bipartite graphs from scratch, but the OR-Library instances have the advantage of being already publicly available.

3. Experimental results

This section presents some preliminary results on the new instances proposed. We have conducted some experiments to determine if they are indeed hard to be solved.

Table I. Dimensions of the bipartite instances before and after preprocessing.

Instance	before preprocessing			after preprocessing		
	$\|V\|$	$\|E\|$	$\|X\|$	$\|V\|$	$\|E\|$	$\|X\|$
bipe2u	550	5013	50	550	5013	50
bip42u	1200	3982	200	990	3610	200
bip62u	1200	10002	200	1199	10000	200
bip52u	2200	7997	200	1819	7326	200
bipa2u	3300	18073	300	3140	17795	300
bipe2p	550	5013	50	550	5013	50
bip42p	1200	3982	200	990	3619	200
bip62p	1200	10002	200	1199	10000	200
bip52p	2200	7997	200	1819	7336	200
bipa2p	3300	18073	300	3140	17797	300

One of the most important practical techniques for the solution of SPG instances is the preprocessing phase, which consists of applying tests that try to eliminate some edges or vertices from the instance before any optimization algorithm is invoked. The hypercube and code covering instances could not be reduced, either by the traditional reduction tests of Duin and Volgenant (1989) or by the recent tests proposed by Uchoa et al. (1999). The sparse bipartite instances can be slightly reduced by some very simple tests. For instance, in the case of instance depicted in Figure 4, vertex 2 and its incident edge can be removed by the NTD1 Non-Terminal Degree 1 test. In the same figure, vertex 3 could be removed and its incident edges replaced by a single edge with cost equal to 2 from vertex 5 to 7 by the NTD2 test. Table I shows the dimensions (number of nodes, number of edges, and number of terminals) of the bipartite instances before and after preprocessing. Although all tests were applied, only the TD1, NTD1, and NTD2 tests described by Duin and Volgenant (1989) lead to some reductions.

Tables II to IV present the results obtained for each of the new classes of instances. For each instance, we first report its dimensions (number of nodes, number of edges, and number of terminals). Next, we give the weights of the solutions found by six different heuristics: (SPH) shortest-path heuristic (Takahashi and Matsuyama, 1980), (HGP) hybrid GRASP with perturbations (Ribeiro et al., 2001), (HGP-PR) hybrid GRASP with perturbations and adaptive path-relinking (Ribeiro et al., 2001), (RTS) reactive tabu search (Bastos and Ribeiro, 2001), (RTS-PR) reactive tabu search with path-relinking (Bastos and Ribeiro, 2001), and (GHLS3) a variant (Poggi et al., 2001a) of HGP-PR which uses a more powerful local search based on key-nodes and a larger pool of elite solutions. These implementations were chosen only because they were readily available to us. The shortest-path heuristic is included in this study only to indicate to the reader how a simple and fast approximate algorithm performs for these instances. Finally, we give the linear programming lower bound (LP) obtained by solving the linear relaxation of the directed cut formulation (Wong, 1984) and an estimate (gap) of the duality gap, given by the percentual difference between the LP bound and the best (GHLS3) upper bound. Results for the bipartite instances in Table IV were obtained after preprocessing. Results in bold face indicate optimal values obtained by GHLS3 proved by the algorithm described by Poggi et al. (2001b).

We now comment on the relative effectiveness of the heuristics when applied to the new classes of instances. We stress that our goal here is not to try to establish the superiority of one heuristic over the others. Instead, we want to show that the new instances provide a test bed on which insightful conclusions can be easily drawn, which is not always the case for the instances currently available in the literature.

We first compare the five first heuristics: SPH, HGP, HGP-PR, RTS, and RTS-PR. HGP-PR is clearly the literature heuristic with the best performance among them, finding the best solution for all but three instances. Comparing the implementations of metaheuristics which do not make use of path-relinking, HGP is never less effective than RTS for the unperturbed instances, which indicates that it is more suited to instances with many global optima. On the other hand, RTS performs better than HGP for most perturbed instances. Finally, GHLS3 outperforms all other heuristics and finds solutions which are always at least as good as those produced by the best among the first five heuristics.

Table II. Hypercube instances

Instance	$\|V\|$	$\|E\|$	$\|X\|$	SPH	HGP	HGP-PR	RTS	RTS-PR	GHLS3	LP	gap (%)
hc6u	64	192	32	41	39	39	39	39	**39**	37.1	4.87
hc7u	128	448	64	80	77	77	77	77	**77**	73.4	4.68
hc8u	256	1024	128	157	149	149	151	151	148	145.1	$\leq$ 1.96
hc9u	512	2304	256	319	296	296	304	304	292	286.8	$\leq$ 1.78
hc10u	1024	5120	512	627	588	588	606	606	582	567.7	$\leq$ 2.46
hc11u	2048	11264	1024	1219	1173	1173	1200	1200	1162	$\geq$1125.2	$\leq$ 3.17
hc12u	4096	24576	2048	2427	2336	2336	2396	2396	2303	$\geq$2201.1	$\leq$ 4.42
hc6p	64	192	32	4302	4017	4003	4003	4003	**4003**	3867.6	3.38
hc7p	128	448	64	8384	7932	7905	7909	7909	**7905**	7646.8	3.27
hc8p	256	1024	128	16746	15637	15376	15573	15526	15322	15115.7	$\leq$ 1.35
hc9p	512	2304	256	32509	31108	30572	30996	30920	30258	29877.6	$\leq$ 1.26
hc10p	1024	5120	512	64472	61905	61030	61633	61605	60494	59213.4	$\leq$ 2.12
hc11p	2048	11264	1024	128204	123129	120804	122741	122741	120096	117388.7	$\leq$ 2.25
hc12p	4096	24576	2048	253825	244674	243390	244477	244477	238673	$\geq$232709.0	$\leq$ 2.50

The time limit for the computation of the LP bound was set at 6 hours on a 400 MHz Pentium II machine.

Table III. Code cover instances

Instance	$\|V\|$	$\|E\|$	$\|X\|$	SPH	HGP	HGP-PR	RTS	RTS-PR	GHLS3	LP	gap (%)
cc6-2u	64	192	12	32	32	32	32	32	**32**	29.8	6.88
cc3-4u	64	288	8	23	23	23	23	23	**23**	21.0	8.70
cc3-5u	125	750	13	36	36	36	36	36	**36**	$\geq$32.8	$\leq$ 8.89
cc5-3u	243	1215	27	76	72	71	74	73	**71**	69.5	2.11
cc9-2u	512	2304	64	187	171	171	178	171	167	$\geq$162.7	$\leq$ 2.57
cc6-3u	729	4368	76	217	201	198	209	206	197	$\geq$194.1	$\leq$ 1.47
cc3-10u	1000	13500	50	132	126	126	130	129	**125**	123.8	0.96
cc10-2u	1024	5120	135	381	349	346	362	359	342	$\geq$332.4	$\leq$ 2.81
cc3-11u	1331	19965	61	163	154	154	158	157	153	$\geq$151.0	$\leq$ 1.31
cc3-12u	1728	28512	74	191	186	186	187	187	186	$\geq$182.0	$\leq$ 2.15
cc11-2u	2048	11263	244	687	624	619	669	651	614	$\geq$600.5	$\leq$ 2.20
cc7-3u	2187	15308	222	612	562	554	598	588	552	$\geq$534.1	$\leq$ 3.24
cc12-2u	4096	24574	473	1315	1201	1184	1287	1255	1179	$\geq$1141.0	$\leq$ 3.22
cc6-2p	64	192	12	3388	3271	3271	3271	3271	**3271**	3078.3	5.89
cc3-4p	64	288	8	2349	2338	2338	2338	2338	**2338**	2194.0	6.16
cc3-5p	125	750	13	3673	3667	3661	3664	3664	**3661**	$\geq$3384.6	$\leq$ 7.55
cc5-3p	243	1215	27	8266	7491	7404	7484	7484	7299	$\geq$7117.8	$\leq$ 2.48
cc9-2p	512	2304	64	18704	17836	17376	17946	17904	17296	$\geq$16766.0	$\leq$ 3.06
cc6-3p	729	4368	76	22680	20850	20554	21060	20657	20458	$\geq$20064.1	$\leq$ 1.93
cc3-10p	1000	13500	50	14149	13084	13061	13118	13003	12860	$\geq$12663.6	$\leq$ 1.53
cc10-2p	1024	5120	135	38608	36552	35867	37234	36545	35466	$\geq$34297.2	$\leq$ 3.30
cc3-11p	1331	19965	61	17111	15924	15728	15940	15917	15609	$\geq$15436.8	$\leq$ 1.10
cc3-12p	1728	28512	74	20626	19285	19167	19201	19159	18838	$\geq$18634.9	$\leq$ 1.08
cc11-2p	2048	11263	244	70666	66073	64334	68486	67328	63841	$\geq$61905.0	$\leq$ 3.03
cc7-3p	2187	15308	222	63339	59005	57601	61190	60303	57339	$\geq$55071.9	$\leq$ 3.95
cc12-2p	4096	24574	473	135953	126541	122928	131778	128015	121772	$\geq$117884.5	$\leq$ 3.19

The time limit for the computation of the LP bound was set at 6 hours on a 400 MHz Pentium II machine.

4. Concluding remarks

The results in Tables I to IV clearly indicate that the new instances proposed in this work are not amenable to reductions by current preprocessing techniques and that the linear programming upper bounds have large gaps and are hard to be computed. State-of-the-art heuristics, which found optimal solu-

Table IV. Bipartite instances

Instance	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert X\rvert$	SPH	HGP	HGP-PR	RTS	RTS-PR	GHSL3	LP	gap (%)
bipe2u	550	5013	50	60	55	55	55	55	54	52.4	≤ 2.96
bip42u	1200	3982	200	270	248	239	259	258	237	232.0	≤ 2.11
bip62u	1200	10002	200	247	229	227	233	231	221	213.3	≤ 3.48
bip52u	2200	7997	200	275	248	247	261	261	235	229.1	≤ 2.51
bipa2u	3300	18073	300	385	358	356	371	367	342	$\geq$329.3	≤ 3.71
bipe2p	550	5013	50	6334	5721	5684	5687	5666	5660	5515.4	≤ 2.55
bip42p	1200	3982	200	28758	25737	25175	25833	25285	24818	24373.8	≤ 1.79
bip62p	1200	10002	200	25267	23877	23291	23779	23500	22944	22445.2	≤ 2.17
bip52p	2200	7997	200	29616	26137	25415	26079	25884	24936	24186.3	≤ 3.01
bipa2p	3300	18073	300	40457	37368	36439	37415	37345	35774	34685.7	≤ 3.04

The time limit for the computation of the LP bound was set at 6 hours on a 400 MHz Pentium II machine.

tions for almost all instances currently in use, faced much more difficulties for the new instances. Fewer optimal solutions were found and the numerical results are more discriminant, allowing a better assessment of the effectiveness and the relative behavior of different heuristics. The new instances have been available at the SteinLib repository (Koch et al., 2001) since July 2001.

References

M.P. BASTOS AND C.C. RIBEIRO, "Reactive tabu search with path-relinking for the Steiner problem in graphs", in *Essays and Surveys in Metaheuristics* (C.C. Ribeiro and P. Hansen, eds.), pages 39–58 , Kluwer, 2001.

J.E. BEASLEY, "OR-Library: Distributing test problems by electronic mail", *Journal of the Operational Research Society* 41 (1990), 1069–1072.

C.W. DUIN, *Steiner's problem in graphs: Approximation, reduction, variation*, Doctorate Dissertation, Institute of Actuarial Science and Economics, University of Amsterdam, 1994.

C.W. DUIN AND T. VOLGENANT, "Reduction tests for the Steiner problem in graphs", *Networks* 19 (1989), 549–567.

C.W. DUIN AND S. VOSS, "Efficient path and vertex exchange in Steiner tree algorithms", *Networks* 29 (1997), 89–105.

C.W. DUIN AND S. VOSS, "The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs", *Networks* 34 (1999), 181–191.

M. GENDREAU, J.-F. LAROCHELLE, AND B. SANSÓ, "A tabu search heuristic for the Steiner tree problem", *Networks* 34 (1999), 163–172.

T. KOCH, A. MARTIN, AND S. VOSS, "SteinLib Testdata Library", online document at http://elib.zib.de/steinlib/steinlib.html, last visited on August 31, 2001.

J.H. VAN LINT, "Recent results on perfect codes and related topics", in *Combinatorics* (M. Hall Jr. and J.H. van Lint, eds.), pages 163–183, Kluwer, 1975.

T. KOCH AND A. MARTIN, "Solving Steiner tree problems in graphs to optimality", *Networks* 32 (1998), 207–232.

A.P. LUCENA AND J.E. BEASLEY, "A branch and cut algorithm for the Steiner problem in graphs", *Networks* 31 (1998), 39–59.

S.L. MARTINS, P. PARDALOS, M.G. RESENDE, AND C.C. RIBEIRO, "A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy", *Journal of Global Optimization* 17 (2000), 267–283.

M. POGGI DE ARAGÃO, C.C. RIBEIRO, E. UCHOA, AND R.F. WERNECK, "Hybrid local search for the Steiner problem in graphs", in *Extended Abstracts of the 4th Metaheuristics International Conference*, pages 429–433, Porto, 2001.

M. POGGI DE ARAGÃO AND C.C. DE SOUZA, "Upper bounds for minimum covering codes via tabu search", in *Extended Abstracts of the 3rd Metaheuristics International Conference*, pages 359–364, Angra dos Reis, 1999.

M. POGGI DE ARAGÃO, E. UCHOA, AND R.F. WERNECK, "Dual heuristics on the exact solution of large Steiner problems", *Electronic Notes in Discrete Mathematics* 7 (2001).

T. POLZIN AND S. VAHDATI, "Improved algorithms for the Steiner problem in networks", *Discrete Applied Mathematics* 112 (2001), 263–300.

C.C. RIBEIRO AND M.C. SOUZA, "Tabu search for the Steiner problem in graphs", *Networks* 36 (2000), 138–146.

C.C. RIBEIRO, E. UCHOA, AND R.F. WERNECK, "A hybrid GRASP with perturbations for the Steiner problem in graphs", *INFORMS Journal on Computing* 14 (2002), 228–246.

H. TAKAHASHI AND A. MATSUYAMA, "An approximate solution for the Steiner problem in graphs", *Math. Japonica* 24 (1980), 573–577.

E. UCHOA, M. POGGI DE ARAGÃO, AND C.C. RIBEIRO, "Preprocessing Steiner problems from VLSI layout", *Networks* 40 (2002), 38–50.

D.B. WEST, *Introduction to graph theory*, Prentice-Hall, 2000.

R. WONG, "A dual ascent approach for Steiner tree problems on a directed graph", *Mathematical Programming* 28 (1984), 271–287.