A relax and cut algorithm for the Prize Collecting Steiner Problem in Graphs

Alexandre Salles da Cunha[1], Abilio Lucena[2], Nelson Maculan[1], and Mauricio Resende[3]

[1] Universidade Federal do Rio de Janeiro, Programa de Engenharia de Sistemas e Computação, acunha@cos.ufrj.br
[2] Universidade Federal do Rio de Janeiro, Departamento de Administração,
[3] AT&T Labs Research.

Abstract. In this paper we address the problem of finding a minimal weight tree in a undirected graph with non-negative edges costs and non-negative vertex penalties, the Prize Collecting Steiner Problem in Graphs (PCSPG). Using an approach proposed by Beasley [2], we formulate the PCSPG as a restricted minimum forest problem, which is amenable to be solved by a Lagrangean relaxation procedure. In our approach some inequalities are dualized when they first become violated and are dropped when their multipliers are zero. As soon as new stronger lower bounds are found, we temporarily modify the cost of the edges present at the Lagrangean solution, and call an upper bounding procedure based on an approximative algorithm [14]. We report several computational results.

1 Introduction

Consider a undirected graph $G = (V, E)$ with non-negative integer edge costs $\{c_e : \forall e \in E\}$ and with non-negative integer vertices penalties $\{d_i : \forall i \in V\}$. An optimal solution to the PCSPG is a tree in G that has minimal cost: the sum of its edge costs plus the sum of the penalties of the vertices not spanned by the tree.

Many practical problems can be modeled as the PCSPG, for example, the design of a local access telecommunications network. In this particular application, the cost of the edges and the penalties associated with each vertex may represent, respectively, the infra-structure costs to connect specified locations (vertices) and the potential profit loss for leaving a location (customer) out of the coverage.

The first work that seems to be correlated to the PCSPG is the paper of Segev [16], in which a Lagrangean relaxation based Branch and Bound algorithm was proposed to solve the *Single Point Weighted Steiner Tree Problem*. In the latter, one seeks a tree with a pre-specified root and pays penalties for not spanning non-root vertices. The Single Point Weighted Steiner Tree Problem is a variant of the PCSPG, in which one of its vertices has an infinity penalty. Another paper that also considered this variant is [6].

The term *Prize-Collecting* was first introduced by Balas [1] in the context of the Traveling Salesman Problem and, since then, it has been used to designate problems in which there is a clear trade-off between paying the cost to add a new vertex to a solution and paying the penalty for leaving such vertex out of it.

Various approximative algorithms have been proposed to solve the PCSPG (see for instance [3] and [8]). Goemans and Williamson [8] proposed a 2 approximative algorithm that inspired many other works ([4], [14], [9] and [5]).

Recently, Lucena and Resende [13] presented an integer programming formulation to the PCSPG. They also solved to optimality several instances by a linear programming relaxation algorithm that generates cutting planes dynamically.

Here, we adapt the extended formulation of the Steiner Problem in Graphs (SPG) proposed by Beasley [2] to reformulate the PCSPG as the problem of determining a forest with additional constraints at a minimum cost. On the one hand, if we relax these additional constraints in a Lagrangean fashion and if we solve the remaining unrestricted minimum forest problem, we provide lower bounds for the PCSPG. To strengthen these lower bounds, as soon as the Lagrangean solution violates valid inequalities, we generate a cutting plane and relax it in a Lagrangean fashion. In our approach, those dinamically generated inequalities are dropped from the objective function as soon as their Lagrangean multipliers become zero. With this strategy, we are able to deal with families of inequalities that are exponential in number. In some sense, this strategy of generating cutting planes dynamically in a Lagrangean relaxation framework follows a previous paper of Lucena [12]. On the other hand, we derive upper bounds by running a "powered" heuristic procedure based on [14].

This paper is organized as follows: in section 2, we present an Integer Programming formulation for the PCSPG. In section 3, we describe the main ideas of the algorithm and, in section 4, we report computational results. We conclude this work in section 5.

2 An IP Formulation for the PCSPG

To formulate the PCSPG as a minimum forest problem with additional constraints, we add an artificial vertex indexed by 0 together with artificial edges $\{(0,i), \forall i \in V : c_{(0,i)} = d_i\}$. Hereafter, we denote the resulting graph $G_0 = (V_0, E_0)$ as the *expanded graph*. So, the new sets are simply: $V_0 = V \cup \{0\}$ and $E_0 = E \cup \{(0,i), \forall i \in V\}$. One should note that, if we consider the expanded graph, one typical feasible solution for the PCSPG is a forest with two disjoint components (trees). The first component has only artificial edges and its cost is, by construction, the sum of the penalties of the vertices that are connected directly to the artificial vertex 0. The second component is constituted by the remaining set of vertices and a spanning tree connecting them. With this expanded graph, we are seeking a two component forest with minimal cost.

If we define decision variables x_e, $\forall e \in E_0$, assuming the value of one if edge e belongs to the solution, and in addition, defining the set of edges induced by

a subset S as $E(S) := \{e = (i,j) : i,j \in S \subseteq V\}$, the sum of the decision variables over a set of edges M as $x(M) := \sum_{e \in M} x_e,\ M \subseteq E_0$, and, finally, the cut of vertex i as $\delta(i) := \{e : e = (i,j) \in E\}$, we can state our integer programming formulation for the PCSPG as

$$\min \left\{ \sum_{e \in E_0} c_e x_e : x \in \mathcal{R} \cap \mathbb{Z}^{|E|+|V|} \right\} \tag{1}$$

where $\mathcal{R}$:

$$x(E_0) = |V_0| - 2 \tag{2}$$

$$x(\delta(i)) \geq 2(1 - x_{(0,i)}),\ \forall i \in V : d_i = 0 \tag{3}$$

$$x(E(S)) + \sum_{i \in S - \{j\}} x_{(0,i)} \leq |S| - 1,\ \forall j \in S,\ \forall S \subseteq V \tag{4}$$

$$0 \leq x_e \leq 1,\ \forall e \in E_0. \tag{5}$$

Equation (2) imposes the required number of edges of the two component forest. Inequalities (3) states that if $c_e \geq 0, \forall e \in E$, in any optimal solution, vertices with zero penalty must have a degree of at least 2, or must not be spanned. Otherwise, a not worse tree would be found by removing such vertex and its hitting edge from the solution. Finally, inequalities (4), called *generalized subtour elimination constraints* (GSEC's), guarantee that the solution is cycle-free. These inequalities (4) were independently proposed by [7], [12] and [15].

It should be clear that if we relax inequalities (3) and (4) in a Lagrangean fashion, the remaining problem can be seen as the problem of finding a forest with exactly $|V_0|-2$ edges with additional constraints, at a minimum Lagrangean cost. In terms of the shape of the forest, these additional constraints impose conditions to the degree of zero-penalty nodes and that if a vertex $i \in V$ is connected to the artificial one, $x_{(0,i)} = 1$, i must not be connected to any other vertex in V.

3 The relax-and-cut algorithm

3.1 Lower bounding

To derive valid lower bounds for the PCSPG formulation presented in the previous section, we relax inequalities (3) - (4) in a Lagrangean fashion, and we get the Lagrangean Problem

$$LP(u) = \min \left\{ \sum_{e \in E_0} C_e(u) x_e : x \text{ is a forest with } |V_0| - 2 \text{ edges} \right\} \tag{6}$$

where u and $C_e(u)$ denote the vector of Lagrange multipliers and the Lagrangean cost of edge e as a function of the Lagrangean multipliers, respectively. One can

solve (6) for a particular choice of u in polynomial time, for example, by using Kruskal's [10] algorithm. To solve the Lagrangean Dual associated with (6), we use Subgradient Optimization (SO).

Here, instead of the traditional Lagrangean relaxation approach in which all relaxed inequalites are dualized, we adopt a significantly different methodology. Some of the relaxed inequalities, those of type (3) and those of type (4) generated by sets $S : |S| = 2$ are dualized during all the time, no matter if they are violated or not. One should note that we have a huge number of inequalities of type (4), even for instances of moderate sizes. In principle, it should be advantageous not to consider all of them at the same time. Thus, for these inequalities, we proceed in a different way: we explicitly dualize them only when they first become violated. For each solution $x(u)$ of (6) unfeasible to (1), we find sets S that generate violated inequalites. With the expanded graph, we have typically two kinds of solutions that are cutted off by inequalities of type (4). The first one is a solution that has a cycle. Considering that we use Kruskal's algorithm to solve (6), we do not have to deal with this kind of infeasibility: our Lagrangean solution is garanteed to be cycle-free. In the second unfeasible solution, there is a vertex $i \in V$ simultaneously connected to the artificial vertex 0 and to another vertex $k \in V$. Consider this vertex i and vertex j in (4). If a particular set S provides a violated inequality, all subsets of $S' \subseteq S : i \in S'$ also do if we choose j such that $j \neq i$. In our approach we generate violated inequalities by sets S' of maximal cardinality and we choose the vertex j such that $j = \arg\max_{y \in S, y \neq i}\{C_{(0,y)}(u)\}$, where i is, as before, the vertex directed connected to 0.

3.2 Upper bounding

To derive a valid initial upper bound, we run a powered version of Minkoff's Unrooted-Growth-Phase with Global-Strong-Prunning algorithm [14]. At the end of the unrooted growth phase, we have a forest as an output. Each tree in this forest is prunned by the Best-Tree-Prunning procedure [14]. The resulting output tree is then post processed by a minimum spanning tree (MST) algorithm and by an *one-neighborhood local optimization* [4]. Considering that this local optimization is time consuming, we restricted its use only to the first call of our upper bounding strategy.

During the next iterations of the SO, we use dual information to drive our heuristic procedure towards new better upper bounds. As soon as we find new stronger lower bounds, we temporarily set the costs of the edges $e \in E$ at the Lagrangean solution to zero, and we run Minkoff's algorithm plus MST post-processing to this modified instance. With this approach, we make the edges in the Lagrangean solution more atractive to the greedy algorithm and we provide a more diversified set of input data to the heuristic.

3.3 Reduction tests and variable fixing

Before running the SO, we perform reduction tests as described in [13]. During the SO, we estimate linear programming reduction costs $RC_e(u)$ to all edges

in the Lagrangean solution. If $RC_e(u) + Z_l(u) > Z_u$, edge e is surely not in any optimal solution. In the last inequality, $Z_l(u)$ and Z_u are, respectively, the current lower bound and the best upper bound obtained so far. If an edge has been fixed, it is not considered by both Kruskal's algorithm and by the heuristic.

4 Computational Results

In this work, we allowed the SO procedure to perform a maximum number of 10,000 iterations, halving the step size parameter in the subgradient algorithm every 600 iterations without improving the best lower bound.

We tested the described algorithm with 64 instances (34 instances of P and K series of [9] and 30 instances of C and D series of [4]). We sumarize the results in the table below, where it are shown the lower and upper bounds (LB) and (UB) obtained at the end of the 10,000 iterations if optimality was not proved, or else the required number of iterations to prove it. In the next columns, we present duality gaps and the percentual of edges $e \in E$ that we were able to fix.

It is important to note that our heuristic procedure produced solutions of high quality. The heuristic hit the optimal known values for all but 4 intances and, in this case, the maximum deviation from optimal values was about 1.1%. Optimality was proved for 24 out of 64 instances and a maximum duality gap of 11.6% was obtained. It is also important to mention that these results were obtained for the same maximum number of subgradient iterations per halving frequency ratio, even though for some instances a better ratio was found.

5 Conclusions

In this paper, we presented an algorithm that dynamically generates cuts in a Lagrangean relaxation framework for the PCSPG. On the one hand, the heuristic procedure provided solutions of high quality. On the other hand, we have theoretical evidence that the existing duality gaps could be closed to optimality. In a previous work [13], the linear programming relaxation of this formulation lead to no duality gaps for all the instances tested in this work. We conjecture that the gaps we report in this paper may be correlated to a cycling behavior of the Lagrangean solution, probably due to instances symetries. But this point is still an opened question that requires further investigation. We intend to implement cycles of reducing tests and subgradient optimization in order to reduce these gaps.

Instance	$\|V\|$	$\|E\|$	Iterations	LB	UB	$\frac{UB-LB}{LB}$ − %	Opt ?	% of fixed edges of E
C1-A	500	625	15	17.6	18		Yes	
C2-A			10	49.1	50		Yes	
C3-A			3,282	413.3	414		Yes	
C4-A			6,400	617.1	618		Yes	
C5-A				1,077.0	1,080	0.3		7.8
C6-A	500	1000	33	17.3	18		Yes	
C7-A			51	49.5	50		Yes	
C8-A				357.8	362	1.2		14.6
C9-A				530.3	533	0.5		17.0
C10-A				855.3	859	0.4		19.2
C11-A	500	2500	1,992	17.1	18		Yes	
C12-A				36.1	38	5.3		56.8
C13-A				233.0	236	1.3		34.9
C14-A				288.5	293	1.6		19.2
C15-A				496.1	501	1.0		16.2
D1-A	1000	1250	4	18.0	18		Yes	
D2-A			917	49.2	50		Yes	
D3-A				803.5	807	0.4		16.9
D4-A				1,197.7	1,203	0.4		6.7
D5-A				2,153.7	2,157	0.2		6.8
D6-A	1000	2000	20	17.3	18		Yes	
D7-A			217	49.1	50		Yes	
D8-A				752.4	755	0.3		24.0
D9-A				1,066.0	1,070	0.4		11.1
D10-A				1,066.2	1,671	0.3		7.3
D11-A	1000	5000	1,887	17.1	18		Yes	
D12-A				39.2	42	7.1		57.1
D13-A				440.9	445	0.9		26.1
D14-A				593.3	602	1.5		22.8
D15-A				1,037.0	1,042	0.5		17.1
P100	100	317		787,779.0	803,300	2.0		30.3
P100.1		284		885,512.8	926,238	4.6		4.7
P100.2		297	3,392	401.641,0	401,641		Yes	
P100.3		316	4,702	659,644.0	659,644		Yes	
P100.4		284		809,642.1	827,419	2.2		18.0
P200	200	587		1,284,900,5	1,317,874	2.6		15.7
P400	400	1200		2,395,163.3	2,459,904	2.7		3.0
P400.1		1212		2,753,256.1	2,808,678	2.0		6.4
P400.2		1196		2,479,833.3	2,518,577	1.6		18.9
P400.3		1175		2,897,543.5	2,951,725	1.9		6.6
P400.4		1144		2,793,107.1	2,852,956	2.1		3.7
K100	100	351	2,987	135,511.0	135,511		Yes	
K100.1		348	2,021	124,128.0	124,128		Yes	
K100.2		339	7,567	200,262.0	200.262		Yes	
K100.3		407	6,483	115,953.0	115,953		Yes	
K100.4		364	521	87,498.0	87,498		Yes	
K100.5		358	258	119,078.0	119,078		Yes	
K100.6		307	41	132,886.0	132,886		Yes	
K100.7		315	3,156	172,457.0	172,457		Yes	
K100.8		343		207.184.0	210,869	1.8		5.5
K100.9		333	1,371	122,917.0	122,917		Yes	
K100.10		319	30	133,567.0	133,567		Yes	
K200	200	691		301,255.0	329,311	9.3		0
K400	400	1515		330,440.8	350,093	5.9		0
K400.1		1470		453,519.0	490,771	8.2		0
K400.2		1527		432,441.0	477,073	10.3		0
K400.3		1492		390,923.4	415,328	6.2		0.2
K400.4		1426		367,814.8	389,462	5.9		0.1
K400.5		1456		489,988.7	521,310	6,4		0
K400.6		1576		354,624.1	374,849	5.7		0
K400.7		1442		450,231.5	474,466	5.4		0
K400.8		1516		399,460.0	418,614	4.8		0
K400.9		1500		359.604.8	383,105	6.5		0
K400.10		1507		357,092,7	398,618	11.6		0.1

Table 1: Computational results for the C and D instances of [4] and for the P and K instances of [9].

References

1. Balas, E. The prize collecting traveling salesman problem. *Networks*, 19: 621–636, 1989.
2. Beasley, J. An SST-Based Algorithm for the Steiner Problem in Graphs. *Newtworks*, 19: 1-16, 1989.
3. Bienstock, D., Goemans, M.X., Simchi-Levi, D., Williamson, D.. A note on the prize collecting travelling salesman problem. *Mathematical Programming*, 59: 413-420, 1993.

4. Canuto, S.A., Resende, M.G.C., Ribeiro, C.C.. Local search with perturbations for the prize collecting Steiner tree problem in graphs. AT &T Tecnical Report, 1999.
5. Cole, R., Hariharan, R., Lewenstein, M., Porat, E.. A faster implementation of the Goemans-Williamson clustering algorithm. Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-25, 2001.
6. Engewall, S., Gothe-Lundgren, M. Varbrand, P. A Strong Lower Bound for the Node Weighted Steiner Tree Problem. *Networks*, 31: 11-17, 1998.
7. Goemans, M.X.. The Steiner tree polytope and related polyhedra. *Mathematical Programming*, 63: 157-182, 1994.
8. Goemans, M.X., Williamson, D.P.. The primal dual method for approximation algorihms and its applications to network design problems. In D.S. Houchbaum, editor, *Approximation algorithms for NP-hard problems*, 144-191, P.W.S Publishing Co., 1996.
9. Johnson, D.S., Minkoff, M., Phillips, S.. The prize collecting tree problem: Theory and Practice. In *Proc. 11th ACM-SIAM Symp. on Discrete Algoritms*, San Francisco, CA, 2000.
10. Kruskal, J.B.. On the shortest spanning tree of graph and the traveling salesman problem, *Proceedings of the American Mathematical Society*, 7: 48-50, 1956.
11. Lucena, A.. Tight bounds for the Steiner problem in graphs, July 15-17 1991. Talk given at the TIMS-XXX - SOBRAPO XXIII Joint International Meeting, Rio de Janeiro.
12. Lucena, A.. Steiner Problem in Graphs: Lagrangean Relaxation and Cutting Planes. In *Proceedings of NETFLOW93*, 147-154, 1993.
13. Lucena, A., Resende, M.G.C.. Strong lower bounds for the prize-collecting Steiner problem in Graphs. AT & T Technical Report, http://www.research.att.com/mgcr, 2000.
14. Minkoff, M.. The Prize Collecting Steiner Tree Problem. Masther Thesis, Massachusetts Insitute of Technology, 2000.
15. Margot, F., Prodon, A., Liebling, T.M.. Tree polyhedron on 2-tree. *Mathematical Programming*, 63:183-192, 1994.
16. Segev, A.. The Node-Weighted Steiner Tree Problem. *Networks*, 17: 1-17, 1987.