Using Lagrangian Dual Information to Generate Degree Constrained Minimum Spanning Trees

Rafael Andrade[1] Abílio Lucena[2] Nelson Maculan[1]

[1]Universidade Federal do Rio de Janeiro, COPPE-Sistemas, C.P. 68511, 21945-970 Rio de Janeiro, Brasil.

[2] Universidade Federal do Rio de Janeiro, LMQ, Departamento de Administração, Av. Pasteur 250, 22290-240 Rio de Janeiro, Brasil.

Abstract

In this paper we present a Lagrangian based heuristic for the Degree Constrained Minimum Spanning Tree Problem (DSTP). We show how to solve a DSTP instance using only a subset of the original set of edges, thus enabling to tackle problems with thousands of vertices in complete graphs. Local Search integrates the heuristic algorithm, which is an adaptation of the Kruskal's algorithm to deal with vertex degree constraints. Finally, we propose a new classification for DSTP instances, to which we report computational results.

Key words: Lagrangian Heuristics – Local Search – Reduced DSTP.

1 Introduction

Let $G = (V, E)$ be a connected undirected graph with a set V of vertices and a set E of edges. *Costs* $\{c_e \in \mathbb{R} : e \in E\}$ are associated with the edges of G while *degrees* $\{d_i \in \mathbb{N} : i \in V\}$ are associated with its vertices.

A spanning tree $T = (V, E')$ of G is said to be *degree constrained* if no more than d_i tree edges are incident on every vertex $i \in V$. The Degree Constrained Minimum Spanning Tree Problem (DSTP) is to find a least cost degree constrained spanning tree of G. The decision version of DSTP is known to be NP-complete (see Garey & Johnson [6]) and so it is unlikely that a polynomial time algorithm exists for the problem.

Practical applications of DSTP involve, among others, the design of computer, telecommunication and transport networks (see [7], [9],[11],[13],[18] and [19] for details).

The very first exact and and nonexact solution algorithms for DSTP were proposed by Gavish [7] and Volgenant [18] and are based on Lagrangian

Relaxation. Zhou and Gen [19] suggested a Genetic Algorithm for the problem. Narula and Ho [14] proposed a greedy heuristic and an exact solution algorithm based, yet again, on Lagrangian Relaxation. Savelsbergh and Volgenant [16] have also proposed a Lagrangian based exact solution algorithm for DSTP. Craig, Krishnamoorthy and Palaniswami [5] proposed several heuristics for the problem, including one based on the use of neural networks. Souza and Ribeiro [17] proposed a GRASP [15] based heuristic which uses Variable Neighborhood Descent. Finally, Caccetta and Hill [4] suggested a Branch and Cut algorithm where Subtour Elimination Constraints (SECs) are introduced (as they become violated) as cutting planes.

In this paper, a Lagrangian Relaxation of a standard DSTP formulation is used to guide a greedy generation of degree constrained spanning trees. Each one of the trees thus obtained is then subjected to local improvements. Local Search is implemented here for a neighborhood consisting of all feasible spanning trees that differ from the one in hand by exactly one edge. Experiments which restrict candidate spanning tree edges, at the greedy phase of the algorithm, to a low cardinality subset of E have also been conducted. This conveniently chosen subset of edges has shown, for the test bed used in this study, to have a high probability of containing optimal DSTP solutions.

Results obtained from extensive computational testing indicate that the proposed heuristic is competitive with the best in the literature. Furthermore, optimality could be proven for many of the test instances considered since Lagrangian dual bounds matched heuristic primal ones.

This paper is organized as follows. In Section 2 the DSTP formulation used in this study is presented. A Lagrangian Relaxation of that formulation is described is Section 3. In Section 4 a Restricted DSTP, used to obtain initial DSTP solutions, is introduced. The proposed Lagrangian heuristic, consisting of a greedy construction phase followed by Local Improvement, is detailed in Section 5. In Section 6 a classification of DSTP instances is suggested. Computational experiments for our algorithm are reported in Section 7. Finally, the paper is closed in Section 8 with some conclusions and suggestions for future work.

2 Problem Formulation

The closely related problem of finding a Minimum Spanning Tree (MST) Problem of G will be briefly reviewed before a formulation of DSTP is presented. In order to do so, associate variables $x \in \mathbb{R}^{|E|}$ with the edges of G. Variable x_e, $e \in E$, is set to one if edge e is in the chosen spanning tree. Otherwise, x_e is set to zero.

Denote by $E(S) \subseteq E$, where $S \subseteq V$, the set of edges with both end vertices in S. Accordingly, denote by $\delta(i) \subseteq E$, where $i \in V$, the set of edges having i as an end vertex. A description of the convex hull of incidence vectors of spanning trees of G, denoted here by R_0, is

$$\sum_{e \in E} x_e = |V| - 1, \tag{1}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad S \subset V, \tag{2}$$

$$x_e \geq 0, \quad \forall\, e \in E. \tag{3}$$

Constraint (1) states that exactly $|V| - 1$ edges of G must be implied by x (very much as one would expect from a spanning tree of G). Subtour Elimination Constraints (2) guarantee that no cycle is induced by the edges being selected. The problem of finding a MST of G is thus formulated as

$$\min \quad \{\sum_{e \in E} c_e x_e : x \in R_0\}. \tag{4}$$

Classical references for solving (4) are the $O(|V|^2)$ algorithm of Prim and the $O(|E|log|E|)$ algorithm of Kruskal.

Degree constraints on spanning tree vertices can be enforced with inequalities

$$\sum_{e \in \delta(i)} x_e \leq d_i, \quad \forall\, i \in V. \tag{5}$$

Consequently, if one denotes by R_1 the polyhedral region defined by constraints (1)–(3) and (5), a formulation of DSTP is

$$\min \ \{\sum_{e \in E} c_e x_e : x \in R_1 \cap \mathbb{Z}^{|E|}\}. \tag{6}$$

Formulation (6) has been used in virtually every single DSTP paper in the literature. Likewise, most of the exact solution algorithms for DSTP use the Lagrangian Relaxation of (6) that follows.

3 A Lagrangian Relaxation of the DSTP

Assume that one attaches nonnegative multipliers $\lambda \in \mathbb{R}_+^{|V|}$ to inequalities (5) and dualize them in a Lagrangian fashion. A Lagrangian subproblem of (6), namely

$$\min \{ \sum_{e=(i,j) \in E} (c_e + \lambda_i + \lambda_j) x_e - \sum_{i \in V} \lambda_i d_i : x \in R_0\}, \tag{7}$$

would result. Since $\sum_{i \in V} \lambda_i d_i$ is a constant for a given λ, problem (7) is to find a MST of G under edge costs $\{(c_e - \lambda_i - \lambda_j) : e \in E\}$. From the

Lagrangian duality theory, it is straightforward to establish that an optimal solution value $z(\lambda)$ for (7) gives a lower bound on the optimal solution value z of (6).

The best possible (i.e. largest) DSTP lower bound capable of being attained from (7), say $z(\lambda^*)$, is associated with multipliers λ^* and has a value

$$\max_{\lambda \geq 0} \{ \sum_{e=(i,j) \in E} (c_e - \lambda_i - \lambda_j) x_e - \sum_{i \in V} \lambda_i d_i : x \in R_0 \}, \quad (8)$$

Lower bound $z(\lambda^*)$ on z can be obtained through the use of Subgradient Optimization methods. Typically, these methods operate by generating a sequence of multipliers $\lambda^0, \lambda^1, \ldots$, which converges to λ^*.

In this paper the Subgradient Method (SM) of Held, Wolfe and Crowder [10] is used in an attempt to obtain $z(\lambda^*)$. The SM is adapted here, as suggested in Beasley [2], for dealing with a large number of dualized inequalities.

3.1 Updating the Lagrangian Multipliers

Let z_{ub} be a known upper bound on z and denote by $\lambda^p \in \mathbb{R}_+^{|V|}$ the Lagrangian multipliers being used at iteration p of SM. Accordingly, let x^p be an optimal solution to (7) under multipliers λ^p. After computing subgradients $\{s_i^p = \sum_{e \in \delta(i)} x_e^p - d_i : i \in V\}$ for dualized inequalities (4), multipliers have been updated, in our computational experiments, as

$$\lambda_i^{p+1} = \max \{0, \lambda_i^p + t_p s_i^p\}, \; \forall \, i \in V, \quad (9)$$

where $t_p = \alpha_p \frac{((1+\beta) z_{ub} - z(\lambda^p)}{||s^p||^2}$, $\alpha_p \in (0, 2]$, and $0.01 \leq \beta \leq 0.03$. Empirical values for α_p and β are determined accordingly [1].

4 The Reduced DSTP Problem

The Reduced DSTP is a degree constrained minimum spanning tree problem restricted to a subset of the set of edges of the original problem. We order the set of edges E in increasing edge costs, and the idea is to work with an ordered subset $E' \subseteq E$, expected to present $|E'| << |E|$. This subset includes at least the edges needed to obtain a corresponding MST, once relaxed the degree constraints. We decide how many ordered edges to add to that subset, which is an empirical task that depends on the class of DSTP instance being considered. For instance, we observed that to solve Hamiltonian path problems, we needed an ordered subset E' with a larger percentage of the edges in E than for all other classes of DSTP instances.

This approach allowed to achieve a considerable reduction on the computational time and to determine upper bounds of good quality with a small effort.

The idea is as follows. Consider that the set of edges E is ordered by increasing values of the edge costs. Suppose that we explored k ordered edges, from the set $E = \{e_1, e_2, \cdots, e_k, e_{k+1}, \cdots, e_{|E|}\}$, to determine the corresponding MST. We obtain the reduced set E' by determining empirically an ordered set of $p \leq |E| - k + 1$ edges, and setting $E' = \{e_1, e_2, \cdots, e_k\} \cup \{e_{k+1}, \cdots, e_p\}$, with $E' \subseteq E$. Let $R_2 := R_1 \cap \{x_e = 0 : e \in E \setminus E'\}$. The reduced problem is

$$\min\{\sum_{e \in E'} c_e x_e : x \in R_2 \cap \mathbb{Z}^{|E|}\}. \tag{10}$$

5 A Lagrangian Based DSTP Heuristic

The basic idea behind a Lagrangian heuristic is to attempt to drive dual solutions into primal feasibility. We somewhat generalize this concept by bringing stand alone primal heuristics into the picture. In the proposed framework, Lagrangian Relaxation solutions are repeatedly used to modify input costs for primal heuristics. An alternative approach which uses Linear Programming Relaxation solutions (instead of Lagrangian Relaxation ones) is proposed in [13] for the Steiner Problem in Graphs. In either case one uses dual information to guide the construction of primal feasible solutions.

The very first ingredient in our Lagrangian heuristic is a Kruskal's style greedy procedure. This is used to generate initial DSTP solutions. Typically, the procedure is called for the different dual solutions obtained along the application of the Subgradient Method. For every call, edge costs $\{c_e : e \in E\}$ are modified so that the edges used in the dual solution in hand are made more attractive to be chosen in the primal procedure. Feasible DSTP solutions thus obtained are then submitted to Local Search. Each of the basic ingredients outlined above are described in detail next.

5.1 Greedy heuristic

A greedy heuristic which follows directly from the MST algorithm of Kruskal is used here to obtain initial feasible solutions for DSTP. In this respect, the procedure is initiated with the $|V|$ isolated components formed by the vertices of G. Edges $\{c_e : e \in E\}$ are ordered in increasing value of their costs and the resulting (ordered) list of edges scanned.

More precisely, a relaxed problem of the reduced problem (10) is used to obtain feasible solutions for the problem (6). The Lagrangian heuristic is composed of a Kruskal [12] algorithm, used to determine MST solutions for the relaxed problem; and a heuristic algorithm (HA) used to determine feasible solutions for (6); and a local search procedure, which tries to improve feasible solutions. The relaxed problem (7), defined for the complete set of edges, is used mainly to obtain lower bounds on (6).

The heuristic algorithm used to determine feasible solutions is an adaptation of the Kruskal algorithm, where we incorporated procedures to deal with the vertex constraints (5). The way we define the edge costs to be used with the heuristic algorithm, distinguishes the Lagrangian heuristic [1]. Indeed, if we use Lagrangian costs $\bar{c}_{ij} = c_{ij} + \lambda_i + \lambda_j$, as input for the heuristic algorithm, we have the basic Lagrangian heuristic algorithm. Instead, if we define complementary costs $\bar{c}_e = c_e(1 - \bar{x}_e)$, where $\bar{x}_e = 1$ if edge e is in the Lagrangian solution, and $\bar{x}_e = 0$, otherwise; we have the Lagrangian complementary heuristic algorithm. See Andrade [1] for other approaches. Feasible solution values are given by the original edge costs. A feasible solution can be improved by a local search procedure, which is incorporated in the heuristic algorithm below.

LOCAL SEARCH PROCEDURE (Input: a DSTP T, set $T_m := T$)
for all $e \in T$ **do**

> **Step 1:** Remove e from T, obtaining two components S1 and S2.
> **Step 2:** Let $\bar{e} \in E \setminus E(T)$ be the edge with minimum cost $c_{\bar{e}}$ connecting S1 to S2 with the exclusion of e, respecting the degree constraints, thus obtaining a new DSTP $\bar{T}$, with $E(\bar{T}) = E(S1) \cup E(S2) \cup \{\bar{e}\}$.
> **Step 3:** **if** $(c_{\bar{e}} - c_e < 0)$ and $(\sum_{e \in \bar{T}} c_e < \sum_{e \in T_m} c_e)$ **then** $T_m := \bar{T}$.
> **Step 4:** Restore T.

Output: return T_m.
End

LAGRANGIAN HEURISTIC PROCEDURE (Input: problem (P))
PART 1: REDUCED PROBLEM (RP) SOLUTION

> **Step 1:** Define the set of edges of RP;
> **Step 2:** Define the type of edge costs to be used in the HA;
> **Step 3:** BestLB $:= -\infty$; i := 0; $\lambda^i := 0$;
> **Step 4:** Determine a feasible solution T by the HA;
> **Step 5:** BestUB := UB(T);
> **Step 6:** **while** (BestUB - LB ≥ 1 **and** $i <$ MAXITER1) **do**
>
> > **Step** $i.1$**:** LB := Kruskal(λ);
> > **Step** $i.2$**:** **if**(BestUB - LB ≥ 1 **and** LB $\geq$ BestLB) **then**
> >
> > > **Step** $i.2.1$**:** BestLB := LB;
> > > **Step** $i.2.2$**:** Actualize the edge costs of the RP;
> > > **Step** $i.2.3$**:** Determine a new feasible solution T by the HA;
> > > **Step** $i.2.4$**:** $\bar{T}$:= LocalSearchProcedure(T);
> > > **Step** $i.2.5$**:** BestUB := $\min\{$UB($\bar{T}$),BestUB$\}$;
> >
> > **Step** $i.3$**:** Determine a new set of multipliers λ^{i+1}; $i++$;

PART 2: ORIGINAL PROBLEM SOLUTION

Step 7: Restore the original set of edges;
Step 8: BestLB := $-\infty$; $k := 0$;
Step 9: **while** (BestUB - LB ≥ 1 **and** $k <$ MAXITER2) **do**

Step $k.1$**:** LB := Kruskal(λ), with associated MST T;
Step $k.2$**:** **if**(BestUB - LB ≥ 1 **and** LB $\geq$ BestLB) **then**

Step $k.2.1$**:** BestLB := LB;
Step $k.2.2$**:** **if** T is feasible **then** BestUB := $\min\{$UB(T),BestUB$\}$;

Step $k.3$**:** Determine a new set of multipliers λ^{k+1}; $k++$;

End

In the first part of the Lagrangian algorithm above, *the solution of the reduced problem*, we define the set of edges of the reduced problem, as well as the type of edge costs to be considered in the heuristic algorithm (HA). The HA determines the first heuristic solution T for the problem, considering the modified edges costs. The original cost of a feasible solution T is given by UB(T). We iterate Step 6 until to obtain an optimal solution (i.e. BestUB - LB < 1, once the edge costs are integer), or to reach the maximum number of subgradient iterations for the reduced problem. From Steps $i.2.1$ to $i.2.5$, for each set of Lagrangian multipliers λ^i leading to an improvement on LB, we determine a new feasible solution by the HA. Applying local search to that solution, we can obtain a new upper bound on the optimal solution.

In the second part, we determine a lower bound on the optimal solution of the original problem. We restore the original set of edges; and we solve the relaxed problem as in the first part. Upper bounds are obtained only at feasible solutions.

The heuristic algorithm and the local search procedure constitute the kernel of the Lagrangian heuristic procedure. For small instances, we can apply it at every iteration in step $i.2$ of the algorithm without to penalize the execution time, instead of applying them only when LB $\geq$ BestLB.

6 Classes of DSTP

In [5, 16] we find some classes of DSTP instances. They are classified as Euclidian and non Euclidian instances. Euclidian instances were proved to be of easy solution. Non Euclidian instances, as the **shrd** ones [5], showed to be more complicated. Nevertheless, they proved easy for our algorithm. The classification used in the literature depends on the difficulties of proposed algorithms in solving them. In this sense, according to the experiments performed here, we propose that this which makes an instance hard or not to deal with, are exclusively the vertex degree constraints. Based on the results of our experiments, we made the following observations about the difficulties in solving DSTP instances.

1. **Easy** instances are those where the set of vertices has degree constraints $d_i \leq M$, with $4 \leq M \leq |V| - 1$. Observe that we do not impose all d_i's to be equal and leaves (vertices with $d_i = 1$) may arise;

2. **Medium** instances are those where all vertices have degree constraints $d_i \in \{1, 2, 3\}$.

3. **Hard** instances are those where the set of vertices has degree constraints $d_i \leq 2, \forall\, i \in V$. In this case, we have the Hamiltonian paths and there may be at most two leaves.

7 Computational Results

The algorithms were implemented in C++ and tested on random instances [1] and on the **shrd** instances [11]. We report results using the Lagrangian heuristic with complementary costs. The random instances were carried out on a SUN Ultra1 workstation. The shrd instances were carried out on a workstation HP 900-735 (HP-UX 10.20). We refer to Andrade [1] for a more detailed set of experiments with other classes of instances and using different solution frameworks for the Lagrangian heuristic.

In next tables, we distinguish the results of the reduced problem from those of the original problem. The legend is as follows. N is the number of vertices, $Krus$ is the value of the first feasible solution for the problem; LB and UB are, respectively, a lower bound and an upper bound on the optimal solution. $Iter$ is the number of iterations used to solve each problem. The maximum number of subgradient iterations for the reduced problem solution was 300, and 500 iterations for the original problem relaxation; CPU is the execution time in (minutes : seconds); $Gap = \frac{100(UB-LB)}{LB}$ is the difference, in percentage, between LB and UB (*optm* means optimum values).

In tables 1, 2 and 3 we present results for medium instances. Instances up to 300 vertices have medium gap of 0.005%, with 63% of them being solved to optimality. Instances of 400 and 2000 vertices have, respectively, an average gap of 0.17% and 0.52%. In table 5 we report the mean gap for the remainder medium instances reported in [1], grouped by the same number of vertices. The instances with 2000 vertices presented the largest gaps. The global gap was, in average, 0.188%. For instances up to 400 vertices, the small gaps indicate, in practice, that the proposed solutions must differ from the optimal one in few edges. Thus it is possible to apply a branch and bound algorithm trying to improve these solutions to optimality. We applied the hard procedures (HP), composed of the heuristic algorithm and of the local search procedure, every iteration of the Lagrangian heuristic. Concerning the gaps of both reduced and original problems, they were closed each other; and generally the optimal solution of the reduced problem was the global one.

Table 1: Medium instances (up to 400 vertices) - HP applied every iteration.

	REDUCED PROBLEM						ORIGINAL PROBLEM				
N	Krus	LB	UB	Iter	CPU	Gap%	LB	UB	Iter	CPU	Gap%
100	3815	3789.709	3790	15	0:03	optm	3789.709	3790	1	0:06	optm
100	3858	3829.000	3829	21	0:04	optm	3829.000	3829	1	0:08	optm
100	3983	3915.118	3916	143	0:42	optm	3915.118	3916	1	0:42	optm
100	3913	3879.000	3879	142	0:44	optm	3879.000	3879	1	0:44	optm
100	3836	3836.000	3836	1	0:00	optm	3836.000	3836	1	0:00	optm
100	3872	3839.958	3844	300	3:20	0.104	3839.956	3844	500	3:58	0.104
100	4280	4139.141	4145	300	2:58	0.121	4138.694	4145	500	4:04	0.145
100	3822	3700.804	3709	80	0:05	0.216	3708.024	3709	116	0:14	optm
100	4258	4193.402	4194	169	1:30	optm	4193.402	4194	1	1:30	optm
200	5373	5316.000	5316	18	0:31	optm	5315.830	5316	17	0:48	optm
200	5765	5645.826	5647	300	11:56	0.018	5639.991	5647	500	21:28	0.124
200	5754	5697.289	5698	209	11:34	optm	5697.289	5698	1	11:37	optm
200	5615	5528.786	5531	300	9:32	0.036	5527.198	5531	500	12:53	0.054
200	5609	5491.452	5494	300	11:53	0.036	5490.699	5494	500	15:12	0.055
200	5457	5405.104	5406	300	7:58	optm	5405.104	5406	1	8:00	optm
200	5510	5466.000	5466	45	0:46	optm	5465.212	5466	57	1:11	optm
200	5369	5332.072	5333	53	0:19	optm	5332.046	5333	96	0:46	optm
200	5719	5675.000	5676	174	7:03	optm	5675.000	5676	1	7:05	optm
300	6498	6473.615	6477	300	20:30	0.046	6474.982	6477	500	25:25	0.031
300	6894	6802.476	6829	300	27:10	0.382	6802.463	6829	500	31:05	0.382
300	6454	6427.146	6431	300	16:24	0.047	6429.865	6431	500	22:03	0.016
300	6435	6364.126	6365	189	9:44	optm	6364.126	6365	1	9:49	optm
300	6705	6605.835	6610	300	22:52	0.061	6605.811	6610	500	32:53	0.061
300	6728	6616.262	6617	212	19:19	optm	6616.448	6617	1	27:21	optm
300	6437	6368.602	6369	189	18:28	optm	6368.602	6369	1	18:34	optm
300	6799	6733.301	6734	195	14:49	optm	6733.489	6734	1	33:08	optm
300	6928	6786.879	6821	300	19:33	0.501	6801.437	6821	500	24:12	0.279
400	7459	7413.458	7416	300	40:21	0.027	7413.968	7416	500	49:18	0.027
400	7870	7774.779	7797	300	89:10	0.283	7774.779	7797	500	98:19	0.283
400	7706	7601.189	7609	300	56:34	0.092	7600.809	7609	500	68:30	0.105
400	7641	7534.694	7545	300	44:41	0.133	7534.637	7545	500	54:50	0.133
400	7813	7676.031	7697	300	135:14	0.261	7681.446	7697	500	143:51	0.195
400	7816	7725.502	7758	300	37:36	0.414	7725.281	7758	500	45:31	0.414
400	7879	7711.649	7626	300	38:20	0.182	7711.649	7726	500	49:25	0.182
400	7703	7551.423	7557	300	62:23	0.066	7551.310	7557	500	70:55	0.066
400	7763	7656.812	7666	300	53:31	0.118	7655.914	7666	500	81:32	0.131

Table 2: Medium instances (up to 800 vertices) - HP applied every iteration.

REDUCED PROBLEM							ORIGINAL PROBLEM				
N	Krus	LB	UB	Iter	CPU	Gap%	LB	UB	Iter	CPU	Gap%
500	8290	8269.777	8279	300	91:22	0.109	8269.777	8279	500	100:56	0.109
500	8433	8391.328	8397	300	71:41	0.060	8391.039	8397	500	85:39	0.060
500	8639	8501.562	8502	261	61:07	optm	8501.562	8502	1	61:21	optm
500	8801	8685.362	8715	300	38:21	0.034	8685.545	8715	500	51:31	0.334
500	8707	8589.855	8604	300	93:23	0.163	8589.855	8604	500	118:48	0.163
500	8477	8350.762	8354	300	108:53	0.036	8350.052	8354	500	172:57	0.036
500	8381	8297.015	8343	300	61:31	0.542	8296.384	8343	500	73:03	0.554
500	8558	8427.810	8436	300	75:54	0.095	8427.602	8436	500	86:19	0.095
500	8433	8270.887	8278	300	60:14	0.085	8268.081	8278	500	81:18	0.109
600	9065	9032.571	9038	300	84:18	0.044	9035.000	9038	500	110:07	0.033
600	9407	9309.266	9337	300	22:46	0.290	9308.518	9337	500	38:40	0.301
600	9479	9327.270	9351	300	118:24	0.247	9326.415	9351	500	135:10	0.257
600	9288	9166.877	9192	300	75:36	0.273	9166.586	9192	500	95:56	0.273
600	9459	9362.787	9382	300	80:32	0.203	9362.144	9382	500	105:06	0.203
600	9212	9038.883	9080	300	174:28	0.454	9038.554	9080	500	192:15	0.454
600	9288	9199.385	9203	300	178:28	0.033	9198.622	9203	500	204:41	0.043
600	9432	9276.281	9296	300	121:36	0.205	9276.278	9296	500	150:01	0.205
600	9589	9466.964	9474	300	81:31	0.074	9466.716	9474	500	98:35	0.074
700	9814	9774.235	9787	300	128:52	0.123	9752.602	9787	500	149:27	0.349
700	10240	10108.275	10128	300	146:47	0.188	10105.169	10128	500	169:48	0.218
700	10210	10084.817	10100	300	100:13	0.149	10084.564	10100	500	123:46	0.149
700	10125	9984.396	9992	300	90:49	0.070	9880.210	9992	500	125:33	0.110
700	9981	9906.320	9918	300	183:44	0.111	9606.271	9918	500	210:50	0.111
700	10182	10006.037	10038	300	157:50	0.310	10006.037	10038	500	184:38	0.310
700	10071	9909.907	9922	300	175:24	0.121	9909.908	9922	500	197:02	0.121
700	10039	9924.268	9934	300	177:11	0.091	9924.132	9934	500	199:53	0.091
700	10011	9872.149	9878	300	206:26	0.051	9870.920	9878	500	269:06	0.071
800	10366	10331.113	10341	300	193:36	0.087	10323.527	10341	500	215:46	0.165
800	10529	10324.815	10335	300	270:11	0.097	10324.845	10335	500	310:34	0.097
800	10673	10532.718	10561	300	275:09	0.266	10532.617	10561	500	302:55	0.266
800	10944	10783.407	10785	300	311:59	0.009	10783.248	10785	500	311:59	0.009
800	10557	10431.943	10441	300	309:45	0.086	10431.887	10441	500	345:53	0.086
800	11030	10849.935	10863	300	170:23	0.120	10849.955	10863	500	206:19	0.120
800	10772	10590.755	10609	300	193:49	0.170	10590.334	10609	500	237:55	0.170
800	10909	10737.268	10767	300	233:06	0.270	10337.180	10767	500	267:42	0.270
800	10915	10753.212	10818	300	315:33	0.595	10748.479	10818	500	348:49	0.642

Table 3: Medium instances (up to 2000 vertices) - HP applied every iteration.

REDUCED PROBLEM							ORIGINAL PROBLEM				
N	Krus	LB	UB	Iter	CPU	Gap%	LB	UB	Iter	CPU	Gap%
900	10938	10916.661	10923	300	327:59	0.055	10917.991	10923	500	358:21	0.046
900	11443	11247.338	11248	231	333:06	optm	11247.338	11248	1	333:58	optm
900	11345	11147.713	11181	300	133:42	0.296	11142.109	11181	500	165:44	0.341
900	11222	11104.286	11121	300	230:32	0.144	11104.182	11121	500	279:30	0.144
900	11363	11227.781	11249	300	168:44	0.187	11226.833	11249	500	220:52	0.196
900	11628	11449.773	11478	300	278:22	0.245	11449.671	11478	500	329:06	0.245
900	11462	11345.238	11383	300	205:06	0.326	11345.240	11383	500	241:09	0.326
900	11351	11224.871	11241	300	194:20	0.143	11224.847	11241	500	238:28	0.143
900	11442	11286.264	11330	300	424:45	0.381	11286.299	11330	500	459:37	0.381
1000	11432	11406.127	11423	300	464:29	0.140	11405.920	11423	500	508:33	0.149
1000	11833	11638.829	11731	300	522:51	0.790	11636.939	11731	500	568:44	0.808
1000	11791	11643.912	11698	300	260:54	0.464	11643.860	11698	500	303:29	0.464
1000	11885	11727.745	11767	300	387:55	0.333	11726.565	11767	500	438:41	0.341
1000	12095	11908.857	11958	300	291:12	0.411	11908.738	11958	500	332:21	0.411
1000	11896	11763.414	11811	300	373:27	0.400	11761.809	11811	500	433:00	0.417
1000	12124	11985.336	12027	300	334:37	0.342	11985.336	12027	500	440:39	0.342
1000	11986	11796.509	11862	300	454:46	0.551	11796.249	11862	500	504:30	0.551
1000	11877	11739.316	11804	300	236:35	0.545	11739.271	11804	500	276:05	0.545
2000	15720	15663.389	15718	300	2771:16	0.345	15669.940	15718	500	3013:07	0.319
2000	16414	16238.792	16320	300	1640:18	0.499	16238.351	16320	500	1789:40	0.499
2000	16899	16666.885	16752	300	1064:50	0.510	16664.822	16752	500	1227:14	0.522
2000	16599	16367.061	16496	300	2141:50	0.782	16367.017	16496	500	2287:31	0.782
2000	16802	16519.270	16598	300	4619:01	0.472	16519.270	16598	500	4786:08	0.472

Table 4: Medium instances - HP applied some iterations.

REDUCED PROBLEM							ORIGINAL PROBLEM				
N	Krus	LB	UB	Iter	CPU	Gap%	LB	UB	Iter	CPU	Gap%
2000	15720	15551.000	15720	300	7:10	1.080	15669.940	15720	500	156:27	0.319
2000	16414	16238.792	16325	300	171:13	0.530	16238.351	16325	500	326:15	0.530
2000	16899	16666.885	16762	300	112:16	0.570	16664.822	16762	500	277:45	0.582
2000	16599	16367.061	16515	300	233:17	0.898	16367.017	16515	500	389:29	0.898
2000	16802	16519.293	16596	300	437:54	0.460	16519.266	16596	500	621:36	0.460

Table 5: Mean gaps.

N	Mean %
100	0.028
200	0.026
300	0.085
400	0.171
500	0.162
600	0.205
700	0.170
800	0.203
900	0.202
1000	0.448
2000	0.519
Aver.	0.188

In table 4 we report other results for those instances with 2000 vertices in table 3, but applying the hard procedures only in a global improvement of the lower bound. We observed that, in general, the quality of the solution decreased, but the saved execution time was considerable. In table 3 we needed some days of execution time, while in table 4 we needed only few hours of CPU time. Bear in mind that if we apply directly the algorithm to the original problem, we could need months of CPU time. For instance, a problem with 2000 vertices has 1.999.000 edges and the reduced problems of our instances have, respectively, 8.988, 15.363, 12.669, 19.733 and 33.276 edges. This indicates why we achieved good solution quality in a reasonable CPU time for those large instances.

Table 6 presents the results of the **shrd** instances [11]. We used the same set of instances as in [17]. The best-known solutions in the literature for these instances are reported in column *BestLit* [17]. We reported the solution of the original problems using a single run of the algorithm for all instances we tested. All instances were solved to optimality as showed in table 6, even those with $d_i = 2$ that we introduced for evaluation purposes of the difficulty in solving these instances as Hamiltonian paths problems.

Finally, we present the Hamiltonian paths results in table 7. They were hard to deal with using our algorithm [1]. This class of DSTP instances presented the greatest gaps. The average gap was 6.232%, which is huge in comparison with those reported in the other tables. Comparing with instances of same dimension in table 1 and 2, this class required larger CPU times.

8 Conclusions

This paper presents computational results for the DSTP that improve on those reported in the literature. A more detailed study is reported in Andrade [1]. We introduced a new approach on how to solve large DSTP instances in complete graphs using a reduced problem in a reasonable execution time. The Lagrangian heuristic showed to be very efficient in obtaining optimal solutions for the shrd instances [11, 17]. An exact branch and bound solution framework is being implemented to improve the solutions obtained by the Lagrangian based heuristic algorithm.

References

[1] Andrade, R.C., *Lagrangian Heuristics for the DCMST Problem*, M.Sc. Dissertation, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 1999.

Table 6: SHRD instances.

$shrd_{d_i}$	Krus	LB	UB	BestLit	Iter	CPU	Gap %
150_2	995	894.286	895	*	218	0:02	optm
150_3	700	581.078	582	582	276	0:04	optm
150_4	508	429.233	430	430	128	0:02	optm
150_5	375	338.698	339	339	043	0:00	optm
159_2	1158	903.392	904	*	242	0:02	optm
159_3	679	596.463	597	597	176	0:03	optm
159_4	464	429.047	430	430	144	0:02	optm
159_5	355	331.109	332	332	039	0:01	optm
200_2	2044	1678.042	1679	*	376	0:06	optm
200_3	1245	1087.121	1088	1091	232	0:07	optm
200_4	885	801.420	802	803	136	0:04	optm
200_5	685	627.000	627	632	229	0:07	optm
209_2	2535	1697.199	1698	*	370	0:06	optm
209_3	1268	1091.130	1092	1096	294	0:09	optm
209_4	941	798.684	799	800	199	0:06	optm
209_5	714	628.353	629	629	111	0:04	optm
258_2	3488	2702.097	2703	*	358	0:09	optm
258_3	1921	1744.088	1745	1746	230	0:15	optm
258_4	1369	1275.354	1276	1277	167	0:10	optm
258_5	1099	998.419	999	999	245	0:15	optm
259_2	3683	2713.157	2714	*	358	0:10	optm
259_3	2027	1756.000	1756	1757	262	0:16	optm
259_4	1363	1291.054	1292	1300	133	0:13	optm
259_5	1078	1015.268	1016	1019	112	0:06	optm
300_2	5284	3991.198	3992	*	403	0:16	optm
300_3	2921	2591.804	2592	2595	321	0:31	optm
300_4	2158	1904.314	1905	1907	291	0:28	optm
300_5	1571	1503.168	1504	1504	168	0:16	optm
309_2	5104	3989.178	3990	*	403	0:17	optm
309_3	2847	2584.009	2585	2587	186	0:19	optm
309_4	2090	1897.163	1898	1899	220	0:21	optm
309_5	1562	1473.511	1474	1478	122	0:11	optm

*No references.

Table 7: Hamiltonian paths - HP applied every iteration.

REDUCED PROBLEM						ORIGINAL PROBLEM			
N	Krus	LB	UB	CPU	Gap%	LB	UB	CPU	Gap%
100	4779	4273.356	4432	13:03	3.697	4273.337	4432	13:41	3.697
100	4455	3873.994	3884	22:11	0.258	3873.994	3884	23:01	0.258
100	4743	4418.498	4743	1:20	7.332	4418.497	4743	2:12	7.332
200	6441	5598.935	5856	93:35	4.590	5598.918	5856	95:41	4.590
200	6395	5698.414	5992	35:03	5.141	5698.417	5992	37:43	5.141
200	7209	5970.457	6159	152:51	3.149	5970.146	6159	157:13	3.149
300	8143	6283.000	7531	472:59	19.844	6859.373	7531	477:18	9.781
300	8018	6849.350	7359	223:24	7.431	6849.350	7359	228:21	7.431
300	8053	7086.440	7740	49:22	9.214	7086.322	7740	55:36	9.214
400	9165	7924.928	8401	339:23	5.993	7925.029	8401	347:04	5.993
400	9529	8124.062	8619	362:23	6.080	8124.189	8619	371:17	6.080
400	9083	7736.355	8281	824:35	7.031	7736.256	8281	832:27	7.031
500	10110	8774.070	9282	729:42	5.778	8773.430	9282	742:39	5.790
500	10043	8815.510	9301	404:34	5.501	8815.211	9301	417:24	5.501
500	10455	8586.254	9746	143:38	13.497	8663.615	9746	148:49	12.488

[2] Beasley, J.E., *Lagrangian relaxation*, Personal Notes, Imperial College, London, England, 1992.

[3] Boruvka, O., *Contribution to the solution of a problem of economical construction of electrical networks*, Elektrotechnický Obzr 15, (1926) pp. 153-154.

[4] L. Caccetta and S.P. Hill, *A Branch and Cut Method for the Degree-Constrained Minimum Spanning Tree Problem*, Networks 37(2), (2001) pp. 74-83.

[5] Craig, G., Krishnamoorthy, M., Palaniswami, M., *Comparison of heuristic algorithms for the degree constrained minimum spanning tree*, In: I H Osman and J P Kelly, (editors), Metaheuristics: Theory and Applications, Kluwer, Boston, 1996.

[6] Garey, M.R., and Johnson, D.S., *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[7] Gavish, B., *Topological design of centralized computer networks - formulations and algorithms*, Networks 12, (1982) pp. 355-377.

[8] Geoffrion, A.M., *Lagrangian relaxation for integer programming*, Mathematical Programming Study 2, (1974) pp. 82-114.

[9] Gomes, M.N., Andrade, R.C., Santiago, C.P., Maculan, N., *Spanning Tree Algorithms to Some Hard Combinatorial Problems*, In: Proceedings of OPTIMIZATION DAYS 1997, MONTREAL/CANADA, (1997) pp. 83-84.

[10] Held, M.H., Wolfe, P., Crowder, H.D., *Validation of subgradient optimization*, Mathematical Programming 6, (1974) pp. 62-88.

[11] Krishnamoorthy, M., Ernest, T.A. and Sharaiha, Y.M., *Comparison of Algorithms for the Degree Constrained Minimum Spanning Tree*, Working Paper, CSIRO Mathematical and Information Sciences, Australia, 1998.

[12] Kruskal, J.B., Jr., *On the shortest spanning subtree of a grath and the traveling salesman problem*, Proceedings of the American Mathematical Society 7, (1956) pp. 48-50.

[13] Lucena, A.P., Beasley J.E., *Advances in Linear and Integer Programming*, Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, 1996.

[14] Narula, S.C., Ho, C.A., *Degree constrained minimum spanning tree*, Computers and Operations Research 7, (1980) pp. 239-249.

[15] Resende, M.G.C., *Greedy Randomized Adaptive Search Procedures. Technical report*, Technical report, Information Sciences Research Center, AT&T Labs Research, Florham Park, NJ 07932, 1998.

[16] Savelsbergh, M., Volgenant, T., *Edge exchanges in the degree-constrained minimum spanning tree problem*, Computers and Operations Research 12(4), (1985) pp. 341-348.

[17] Souza, M.C., Ribeiro, C.C., *Variable Neighborhood Descent for the Degree-Constrained Minimum Spanning Tree Problem*, Third Metaheuristic International Conference, Angra dos Reis, (1999) pp. 411-416.

[18] Volgenant, A., *A lagrangean approach to the DCMST problem*, European Journal of Oper. Res. 39, (1989) pp. 325-331.

[19] Zhou, G. and Gen, M., *A note on genetic algorithms for degree constrained spanning tree problems*, Networks 30, (1997) pp. 91-95.