A GRASP heuristic for the Node Degree Constrained Minimum Spanning Tree Problem with Node Degree Costs

Christophe Duhamel, *LIMOS, Université Blaise Pascal, Clermont-Ferrand, France*
Mauricio Souza *DEP, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil*

Keywords: Combinatorial Optimization, network design, constrained spanning tree, metaheuristics

1. Introduction

The Node Degree Constrained Minimum Spanning Tree Problem (NDCMSTP) consists in finding a minimum spanning tree such that the degree of each node is restricted to be not greater than a given threshold. The cost function takes into account two elements: the cost of each selected edge and also the cost associated to the degree of each node. This problem makes sense in telecommunication network design where one has to take the cost of the routers into account. The communication is made through I/O modules whose connectivity, i.e. number of I/O ports is limited. Then in order to match the connectivity requirement, several modules may have to be connected together. Moreover, several modules capacities may be available and the smaller modules may have a cost per connection higher than the larger ones (due to economy of scale). Another point is that the installation cost for routers may be more expensive than the installation cost for terminals as the latter ones do not need any equipement to perform the routing. Therefore, we consider here the node degree cost function as a concave staircase cost function.

Another application for the NDCMSTP is to minimize the delay in high speed networks. In such networks, broadcasting may lead to high delays: the incoming message is copied at each router in order to be sent to each child and one copy is needed for each child. As this operation consumes a lot of time, it is of practical interest to build a communication networks taking into account the degree of each router.

This problem is NP-hard [5], even when the same degree is imposed on every node (k-NDCMSTP). Interestingly, the problem is polynomial on euclidean instances when $k \geqslant 5$.

The NDCMSTP has been well studied in the past. Narula and Ho [11] where the first to propose a branch and bound algorithm. Later, Savelsbergh and Volgenant [14] proposed a branch and bound algorithm using an edge exchange procedure. Gavish [6] proposed a lagrangean relaxation and Volgenant [15] developed a dual-ascent lagrangean relaxation to solve the problem. More recently, Caccetta and Hill [2] proposed a branch and price approach and Cunha and Lucena [3] used a relax and cut. Metaheuristics have been applied to the problem too. Zhou and Gen [16] proposed a genetic algorithm using Prüfer numbers to encode the trees. Krishnamoorthy, Ernst and Sharaiha [9] proposed several heuristics, including a simulated annealing, genetic algorithms and problem space search. Ribeiro and Souza [13] developed a VNS/VND strategy. Raidl and Julstrom [12] presented an evolutionary algorithm using edge set representation and recently Andrade, Lucena and Maculan [1] proposed an effective lagrangean heuristic.

To our knowledge, none of those approaches include a node degree cost function. In this work, we present an hybrid GRASP/VND metaheuristic to compute (hopefuly) good primal solutions to the problem. We also propose a path relinking strategy to further improve the efficiency of our approach. Preliminary numerical results are reported.

2. Mathematical formulation

Let $G = (V, E)$ be a graph, where V is the set of n vertices and E is the set of m edges. To each edge $e \in E$, let $c_e > 0$ be its installation cost. To each vertex $v \in V$, let D_v be the upper limit on its degree d_v. Without

loss of generality, it will be assumed that the restriction is the same for each vertex, that is $D_v = D$ and $F_v(d_v) = F(d_v)$. The node degree cost function $F(d)$ is a concave staircase function as explained before. Let $x_e \in \{0,\ 1\}$ be the decision variable associated to each edge $e \in E$. The problem can then be stated as follows:

$$\text{Min} \quad z = \sum_{e \in E} c_e(x_e) + \sum_{v \in V} F(d_v)$$

s.t.

$$\begin{array}{lll} x_e \in \text{ST} & & (1) \\ \displaystyle\sum_{e \in E(v)} x_e = d_v & \forall v \in V & (2) \\ d_v \leqslant D & \forall v \in V & (3) \\ & & \\ x_e \in \{0,\ 1\} & \forall e \in E & \\ d_v \in \mathbb{N} & \forall v \in V & \end{array}$$

ST denotes the convex hull of the incidence vectors of spanning trees. Thus, the x variables are restricted to define a spanning tree. Therefore, any valid formulation for the spanning tree can be used. Restriction (2) define the degree variables. Constraint (3) set an upper limit on the degree. It should be noted this model is non linear as the function involves non linear terms $F(d_v)$. Mixed integer linear formulations are available for the NDCMSTP, see for instance Gouveia and Moura [8].

3. An hybrid GRASP/VND metaheuristic

The GRASP metaheuristic has been developped by Feo and Resende [4]. It consists in repeated applications of a two-step procedure: a Construction step to produce an initial solution and an Improvement step based on a Local Search to improve it. The Construction step is usualy a randomized heuristic. It builds a feasible solution T by adding one element at a time. At each iteration, let $\mathcal{C}$ be the set of all possible candidates for insertion into the partial solution T and let f be an evaluation function of the interest of a candidate. For each candidate $c \in \mathcal{C}$, $f(c)$ is its value, $\underline{f}$ and $\overline{f}$ are respectively the better and the worst evaluation over $\mathcal{C}$. Then, the candidate for insertion is randomly chosen within the restricted candidate list (RCL), i.e. the list of candidates e whose evaluation $f(e) \in [\underline{f};\ \underline{f} + \alpha(\overline{f} - \underline{f})]$. Thus, the parameter $\alpha \in [0;\ 1]$ controls the greediness of the selection.

```
procedure GRASP_DCMST
    set T* = ∅
    for it = 1, ..., it_max do
        T ← Construct(α)
        T̄ ← Improve(T)
        if F(T̄) < F(T*) then
            T* ← T̄
        end-if
    end-for
    return T*
end-procedure
```

Figure 1: GRASP metaheuristic

For the NDCMSTP, the following constructive heuristic is proposed: first, feasible degrees are assigned to the nodes, then a solution is build according to those degrees. To assign the degrees, let c_v^k be the cost of the k^{th} shortest edge incident to v. Given a node degree d_v, an underestimation of the node v total cost is

$F(d_v)+\sum_{i=1}^{d_v} c_v^i$. Thus, the node v expansion cost can be approximated as $g(v) = F(d_v+1)-F(d_v)+c_v^{d_v+1}$. The procedure to assign degrees to nodes is presented in figure 2.

procedure `Construct_1`
  set $d_v = 1, \quad \forall v \in V$
  repeat
    $a = \min\{g(v), v \in V\}$
    $b = \max\{g(v), v \in V\}$
    $RCL \leftarrow \{v \in V \mid a \leqslant g(v) \leqslant a + \alpha_1(b-a)\}$
    randomly select $v \in RCL$
    set $d_v \leftarrow d_v + 1$
  until $\sum_{v \in V} d_v = 2(n-1)$
end-procedure

Figure 2: degree assignement

Building a tree when the node degrees are fixed is a NP-hard problem. Thus, once the degrees have been assigned, a compatible tree T will been built using a Kruskal-like heuristic. Let e_v be vertex v free degree. Let $D_0 = \{v \in V \mid e_v = 0\}$, $D_1 = \{v \in V \mid e_v = 1\}$ and $D_+ = \{v \in V \mid e_v > 1\}$ be respectively the set of fixed vertices, the set of leaves and the set of non-leaves in the partial tree T. At each iteration, an edge is inserted into T to fix one vertex of D_1 and the sets are updated. The procedure is presented in figure 3.

procedure `Construct_2`
  set $e_v = d_v, \quad \forall v \in V$
  set $D_0 = \emptyset,\ D_1, D_+$
  repeat
    $a = \min\{c_{ij}, i \in D_1,\ j \in D_+\}$
    $b = \max\{c_{ij}, i \in D_1,\ j \in D_+\}$
    $RCL \leftarrow \{(i,j) \in D_1 \times D_+ \mid a \leqslant c_{ij} \leqslant a + \alpha_2(b-a)\}$
    randomly select $(i,j) \in RCL$
    update e_i, e_j, D_0, D_1 and D_+
  until $D_+ = \emptyset$
  connect the last two vertices from D_1
end-procedure

Figure 3: tree building

At the end of the Construction procedure, a feasible tree has been built, provided the graph G is complete. Otherwise, the procedure may end up without being able to connect all the vertices. Therefore, G is transformed into a complete graph by adding artificial edges (the missing ones) with an arbitrarily high cost. This way, the procedure will construct a degree-compatible tree, even if it is infeasible with respect to the edges in the initial graph. No restoration procedure need to be used: the Local Search will try to remove as many artificial edges as possible, since they are too expensive.

The Variable Neighbourhood Descent (VND) is used as Local Search for the Improvement step. The VND was introduced by Mladenović and Hansen [10] and is designed to be the Local Search of the Variable Neighbourghood Search (VNS) metaheuristic. It consists in defining a set $\mathcal{N} = \{\mathcal{N}_1, \ \ldots, \ \mathcal{N}_K\}$ of K neighbourhood structures. Given a solution $s \in \mathcal{S}$, the neighbourhood $\mathcal{N}_k(s)$ is defined as $\mathcal{N}_k(s) = \{s^{'} \in \mathcal{S} \mid d(s, s^{'} = k\}$, where $d(.,.)$ is a user-defined distance (the $||.||_1$ norm for instance). At each iteration, let $s \in \mathcal{S}$ be the current solution and $\mathcal{N}_k$ be the active neighbourhood structure. The best neighbour $s^{'} \in \mathcal{N}_k(s)$ is selected and the state is updated as shown in figure 4. The VND stops when the current neighbourhood is $\mathcal{N}_K(s)$ and when it contains no improving solution.

For the NDCMSTP, the neighbourhood structures are chosen to rely on k-edge exchange moves: k new edges are inserted into the tree T while k others are removed, producing a new tree $T^{'}$. Thus, T and $T^{'}$ differ from $2k$ edges. To take the node degree into account, three sets are defined according to the current tree T: $E_0 = \{(u,v) \in E \setminus T \mid d_u < D,\ d_v < D\}$, $E_1 = \{(u,v) \in E \setminus T \mid d_u = D,\ d_v < D\}$ and

```
procedure VND
    set k ← 1
    set s ← s_0
    repeat
        s' ← N_k(s)
        if f(s') < f(s) then
            s ← s'
            k ← 1
        else
            k ← k + 1
        end-if
    until k ⩽ K
    return s
end-procedure
```

Figure 4: VND local search

$E_2 = \{(u, v) \in E \setminus T \mid d_u = D,\ d_v = D\}$ are respectively the sets of edges whose insertion would violate 0, 1 and both extremities. Only neighourhood structures $\mathcal{N}_1$ and $\mathcal{N}_2$ are used.

In order to improve the quality of the solution found at the end of each GRASP iteration, a path relinking strategy [7] is used. A pool $\mathcal{P}$ of elite solutions is created, initialy $\mathcal{P} = \emptyset$. Let s_1 be the local optimum returned by the VND. An elite solution $s_2 \in \mathcal{P}$ is randomly chosen and moves of the $\mathcal{N}_1$ structure are performed to transform s_1 into s_2. Those moves are biased to favour the inclusion of elements belonging in s_2 and not in s_1. By doing so, the idea is to try to identify interesting solutions on the path from s_1 towards s_2. Let $s_3 \neq s_1, s_2$ be the best solution found in the path. Then, both s_1 and s_3 are candidates for insertion into $\mathcal{P}$. Each solution is inserted if it is better than the worst solution of $\mathcal{P}$ and if does not already belong to it.

4. Preliminary numerical results

The heuristic has been tested on a set of randomly generated instances. The number of nodes has been set to 25, 50 and 100 nodes. For each size, three densities (25%, 50 % and 100% the density of a complete graph) have been used. Three values for the maximum node degree have been used, $D \in \{4, 5, 6\}$. The cost c_e of each edge $e \in E$ is randomly chosen in the interval $[1; 100]$ and the staircase cost function is as follows: $F = [0\ 30\ 30\ 40\ 40\ 40]$. The program has been coded in C and compiled with gcc version 4.0.1. The experiments are run on a pentium IV processor at 800MHz with 1Gb memory and running linux 2.6.12. For each instance, 1000 iterations of GRASP are run and at each iteration α is randomly chosen in $\{1/10, 2/10, \ldots, 9/10, 1\}$. The pool size has been set to 20.

The results are shown on table 1. For each instance, the optimal value z_{IP} is presented when available. For the instance $(50,\ 1225,\ 6)$, the value 2332 is only an upper bound on the optimal value. The columns z_{GRASP} and z_{PR} show the value computed by GRASP and by GRASP using the path relinking strategy. The columns "feas_C" and "feas_V" report the percent feasibility of the solution computed by the Construction step and then optimized by the VND. The CPU time is in seconds. It is only reported for the GRASP with path relinking as it is only slighly higher than the time for GRASP alone.

For small instances, GRASP alone finds the optimal solution on the most sparse graphs. For the other densities, it stays within 1% of the optimal values. Using the path relinking strategy helps finding nearly all the optimal values. For medium instances, GRASP still performs well, especialy when the path relinking is used. The values are still within 1% of the optimal values. For the biggest instances ($n = 100$), no optimal values could be computed using exact methods. However, one can note that the path relinking strategy is quite effective as it always improves the value of the best solution found. As the number of nodes and the density grow, the CPU time quickly increases. This probably means a better stopping criterion (than stopping

n	m	D	z_{IP}	$feas\_C$	z_{GRASP}	$feas\_V$	z_{PR}	$cpu(s)$
25	75	4	1665	0.000	1665	1.00	1665	0.51
		5	1623	0.000	1623	1.00	1623	0.50
		6	1589	0.002	1589	1.00	1589	0.47
	150	4	1372	0.074	1373	1.00	1372	1.06
		5	1311	0.069	1316	1.00	1311	0.87
		6	1282	0.050	1286	1.00	1284	0.94
	300	4	1476	0.745	1477	1.00	1476	2.39
		5	1429	0.685	1432	1.00	1429	2.18
		6	1403	0.619	1413	1.00	1409	2.08
50	300	4	3336	0.001	3337	1.00	3336	9.47
		5	3191	0.000	3191	1.00	3191	8.93
		6	3109	0.000	3113	1.00	3113	8.35
	600	4	3063	0.039	3065	1.00	3064	17.66
		5	2948	0.030	2953	1.00	2948	16.59
		6	2900	0.038	2900	1.00	2900	14.17
	1225	4	2498	0.645	2504	1.00	2500	39.89
		5	2380	0.578	2386	1.00	2383	34.53
		6	$*2332$	0.605	2339	1.00	2336	29.44
100	1250	4		0.000	6316	1.00	6283	86.11
		5		0.000	6093	1.00	6068	76.09
		6		0.000	5973	1.00	5951	73.13
	2500	4		0.067	5456	1.00	5410	186.97
		5		0.068	5225	1.00	5195	173.05
		6		0.049	5126	1.00	5094	158.46
	4950	4		0.768	5503	1.00	5475	580.32
		5		0.704	5278	1.00	5215	525.80
		6		0.668	5125	1.00	5074	359.23

Table 1: results for small, medium and big instances

after 1000 iterations) should be used on GRASP. One interesting fact is that the limit D has a positive impact on the running time: as D increases, the CPU decreases. An explanation is that the problem becomes less constrained and then more moves are available, which is known to help methods like the VND. One can also note that the feasibility of the solution computed by the Construction procedure mostly depends on the density of the instance. For the smallest density ($m = 0.25n^2$), the solution is nearly never feasible in the original graph. For complete graphs ($m = n^2$), the solution is feasible 60% of the time. After the VND, the solution has always been found to be feasible.

References

[1] Andrade, R., Lucena, A., and Maculan, N., "Using Lagrangian dual information to generate degree constrained spanning trees", *Discrete Applied Mathematics* 154, pp. 703–717, 2006.

[2] Caccetta, L., and Hill, S. P., "A branch and cut method for the degree-constrained minimum spanning tree problem", *Networks* 37, pp. 74–83, 2001.

[3] Cunha, A., and Lucena, A., "Lower and upper bounds for the degree-constrained minimum spanning tree problem", in *Proceedings from International Network Optimization Conference* 1, pp. 186–192, 2005.

[4] Feo, T. A., and Resende, M. G. C., "Greedy randomized adaptive search procedures", *Journal of Global Optimization* 6, pp. 109–133, 1995.

[5] Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., 1979.

[6] Gavish, B., "Topological design of centralized networks: Formulations and algorithms", *Networks* 12, pp. 355–377, 1982.

[7] Glover, F., "Tabu search and adaptive memory programing – Advances, applications and challenges", in *Interfaces in Computer Science and Operations Research* (R.S. Barr, R.V. Helgason, and J.L. Kennington, eds.), pp. 1–75, Kluwer, 1996.

[8] Gouveia, L., and Moura, P., "Models for the Node Degree Constrained Minimum Spanning Tree Problem with Node Degree Costs", submitted to the *Proceedings from International Network Optimization Conference*, 2007.

[9] Krishnamoorthy, M., Ernst, A. T., and Sharaiha, Y. M., "Comparison of algorithms for the degree constrained spanning tree", *Journal of Heuristics* 7, pp. 587–611, 2001.

[10] Mladenović,N., and Hansen, P., " Variable Neighborhood Search", *Comps. in Opns. Res.* 24, pp. 1097–1100, 1997.

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

[12] Raidl, G. R., and Julstrom, B. A., "Edge-sets: an effective evolutionary coding of spanning trees", *IEEE Transactions on Evolutionary Computing* 7, pp. 225–239, 2003.

[13] Ribeiro, C. C., and Souza, M. C., "Variable neighborhood search for the degree-constrained minimum spanning tree problem", *Discrete Applied Mathematics* 118, pp. 43–54, 2002.

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

[15] Volgenant, T., "A Lagrangean approach to the DCMST problem", *European Journal of Operational Research* 39 , pp. 325–331, 1989.

[16] Zhou, G., and Gen, M., "A note on genetic algorithms for the degree-constrained spanning tree problem", *Networks* 30, 91–95, 1997.