Heuristics Algorithms for The Degree Constrained Minimum Spanning Tree Problems

L. Caccetta and Wamiliana
Dept. of Mathematics and Statistics,
Curtin University of Technology, Perth, Western Australia
caccetta@maths.curtin.edu.au

Abstract The Degree Constrained Minimum Spanning Tree Problem is concerned with finding, in a given edge weighted graph G (all weights are non-negative), the minimum weight spanning tree T satisfying specified degree restrictions on the vertices. This problem arises naturally in communication networks where the degree of a vertex represents the number of line interfaces available at a terminal (center). Since, apart from some trivial cases, the problem is computationally difficult (NP-complete), a number of heuristics have been proposed. In this paper we propose two new heuristics: one based on the method of Tabu search and other based on a penalty function approach. Our heuristics are implemented and extensively tested on simulated problems. The computational results support our methods.

Keywords: Minimum spanning tree; Tabu search; Degree constrained

1. INTRODUCTION

Graph theoretic concepts have proven useful in studying problems arising in network design and analysis. Here we consider a graph $G = (V,E)$ to be a collection of vertices V representing the nodes/centers in the network together with a set E of edges representing the network connections/links. Our graphs are finite, undirected and have no multiple edges. Thus letting $V = \{1,2,\ldots,n\}$ we can identify the edges of G as unordered pairs (i,j). The weight (cost or distance) of edge (i,j) is denoted by c_{ij}. The c_{ij}'s are assumed to be non negative.

Given a weighted graph G, a fundamental problem in network design is to select a subset of edges so the resulting network is connected and the total weight of the edges selected is as small as possible. This problem is referred to as the **minimum spanning tree (MST) problem**. The MST problem is easily determined by a greedy algorithm such as Kruskal's [1956] and Prim's [1957].

Here we consider the problem of finding a MST satisfying a degree constraint on each vertex. More precisely, the **Degree Constrained Minimum Spanning Tree (DCMST) problem** is to find a minimum spanning tree T of G such that the degree of vertex i in T is at most b_i, $1 \le i \le n$.

Garey and Johnson [1979] showed that, apart from some trivial cases, the DCMST problem is computationally difficult (NP-complete) by reducing it to an equivalent symmetric Traveling Salesman Problem (TSP). Notice that if the degree bound $b_i = 2$, $\forall\ i \in V$, the problem reduces to a TSP. Thus, it is unlikely a polynomial bounded algorithm exists for solving general DCMST problems.

The DCMST problem has been considered by a number of authors and both heuristic and exact methods have been proposed. In this paper we present two new heuristics: one based on the method of Tabu search and other based on a penalty function approach. The Tabu search method has previously not been applied to this problem. The paper is organized as follows: Section 2 gives an integer programming formulation of the problem; Section 3 briefly reviews some of the solution methods available in the literature; Section 4 discusses our new heuristics; and Section 5 presents the computational results based on 2400 random table problems having 10 to 500 nodes.

2. FORMULATION

From the modeling point of view, the DCMST problem can be formulated as a Mixed Integer Linear Programming as follows:

$$Minimise \quad \sum_{i}^{n}\sum_{j}^{n} c_{ij}x_{ij} \tag{1}$$

subject to:

$$\sum_{i,j\in V} x_{ij} = n-1 \tag{2}$$

$$\sum_{i,j\in V'} x_{ij} \leq |V'|-1 \ , \forall \varnothing \neq V' \subseteq V \tag{3}$$

$$1 \leq \sum_{j=1,i\neq j} x_{ij} \leq b_i \quad , \ i = 1,2,...,n \tag{4}$$

$$x_{ij} = 0 \text{ or } 1 \qquad ,1 \leq i \neq j \leq n \tag{5}$$

Constraint (2) ensures that n-1 edges are selected. Constraint (3) eliminates cycles, and constraint (4) ensures that the degree restrictions are met. Note that $x_{ij} = 1$, if edge (i,j) is selected and 0, otherwise.

Caccetta and Hill [2001] modified the above formulation by replacing (2) and (4) with:

$$\sum_{j\in V} x_{ij} - d_i = 0 \qquad , \qquad 1 \leq i \leq n \tag{6}$$

$$\sum_{i=1}^{n} d_i = 2(n-1) \tag{7}$$

$$1 \leq d_i \leq b_i \ , \qquad , \qquad 1 \leq i \leq n \tag{8}$$

This formulation has n additional variables (d_i's) but n fewer constraints. It was noted in Caccetta and Hill [2001] that this formulation was better computationally in a branch and cut procedure.

3. AVAILABLE ALGORITHMS

There are many exact and heuristics algorithms for DCMST available in the literature.

Exact methods for the DCMST problem have been considered by a number of authors. The methods developed include: Langrangean relaxation by Gavish [1982] and Volgenant [1989]; branch and bound by Narula and Ho [1980], Savelsbergh and Volgenant [1985], and Volgenant [1989]; and the branch and cut method by Caccetta and Hill [2001]. The best results reported in literature are those of Caccetta and Hill [2001].

For heuristics, many variations of Prim's and Kruskal's algorithms have been developed, for example, by Narula and Ho [1980] and Caccetta et al [2000]. The idea is to maintain feasibility of the degree during the spanning tree construction. Caccetta et al. [2000] implemented and tested algorithms on problems with up to 500 vertices.
A Genetic Algorithm was proposed by Zhou and Gen [1997]. They use the Prufer [1918] number to uniquely coding the spanning tree. The use of Prufer number has advantages in that any vertex with degree r will appear exactly r-1 times in the Prufer number. In the method they adopt uniform crossover and perturbation mutation operators as the genetic operators, and implemented the algorithm on problems with up to 50 vertices.

Simulated Annealing was proposed by Krishnamoorthy et al. [1998, 2000]. Further, they also proposed a hybrid method called Problem Space Search, which is a blend of the Genetic Algorithm approach and a simple constructive search method. The algorithms were implemented on problems with 30, 50, 70 and 100 vertices.

Boldon et al. [1996] and Deo and Kumar [1997] proposed a method based on iterative refinement procedure called Iterative Refinement. In this method, the construction starts with finding a MST and then the edges incident to a degree violated vertex are penalized, except the smallest one, to maintain connectivity. With the new weighted edges, the process of calculating a MST is repeated, and it continues until a spanning tree without degree violation is found. They implemented this method using parallel computing on a computer with 8192 processors. This can be done because the nature of the algorithm/method, where every vertex can be assigned a processor and the computational process of penalizing edges is independent (non sequential). They solved problems with up to 5934 vertices.

4. NEW APPROACHES FOR DCMST

For computationally difficult problems heuristics are often a good alternative. We propose two new heuristics; one is based on Tabu Search method and on a penalty approach.

4.1 Tabu Search Approach

Tabu search is a *meta-heuristic* that guides a local heuristic search procedure to explore the solution space beyond local optimality. According to Osman and Laporte [1996] a meta-heuristic is defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and

exploiting the search space. Learning strategies are used to structure information in order to find efficiently near-optimal solutions. Tabu search methods are now widely used in many problems, especially combinatorial optimization problems. In order to improve the efficiency of the exploration process, one needs to keep track not only local information (like current objective function value) but also some information related to the exploration process. This systematic use of memory is an essential feature of Tabu search.

In Tabu Search, an important distinction arises by differentiating between short-term memory and long-term memory. The foundations of Tabu Search lie on these two types of memory. Each type of memory has its own special strategies.

Short-term memory keeps track of the most recent solution attributes that have changed during the past. This memory is referred to as *recency-based memory* in Glover and Laguna [1997]. It is exploited by assigning a tabu active designation to selected attributes occurring in solutions recently visited, and then the solutions containing tabu active elements become *tabu*. However, those solutions only hold tabu status or tabu attributes temporarily, not forever. The Tabu tenure is defined as the duration an attribute remains tabu active (usually measured by the number of iterations). Tabu tenure can vary for different types or combinations of attributes and can also vary over different time or stages of the search.

One term that accompanies the short-term memory is *Aspiration Criteria*. The tabu status of a solution is not absolute, but can be overruled if certain conditions are met, and this is expressed in the form of aspiration criteria.

In some applications, the short-term memory components are sufficient to produce good quality solutions. However, in general, Tabu Search becomes stronger by including longer-term memory and its associated strategy.

One fundamental term in long-term memory is *frequency-based memory*. This memory works by introducing penalties and inducements determined by the relative span of time that attributes have belonged to solutions visited by the search, allowing for regional differentiation.

The *transition frequencies* keep track of how often the attributes change while *residence frequencies* keep track of relative durations attributes occur in the solutions generated. *Intensification strategy* is one other important component in long-term memory. This strategy is based on modifying choice rules to encourage move combinations and solution features historically found good. Intensification strategy may also initiate a return to attractive regions to search them more thoroughly.

Tabu search *diversification strategy* as its name suggests, is designed to drive the search into the new regions. This strategy is often based on modifying choice rules to bring attributes into the solutions infrequently used or alternatively, it may introduce such attributes by partially or fully restarting the solution process.

Using the procedure defined by Hertz et al. [1997] the essentials of Tabu search procedure can be stated as follows. We use the notation that S_k denotes the set of available solutions at iteration k, i^* is always the best available solution and $\mathcal{N}(i,k)$ is the "neighborhood" of solution i at iteration k.

Initialization: Set $k=0$ and initial solution $i=S_0$.
Step 1: Let current solution be i. Set $k=k+1$ and generate a subset S^* of solutions in $\mathcal{N}(i,k)$ such that either one of the tabu conditions is violated or at least one of the aspiration criteria holds.
Step 2: Choose the best solution j with respect to the objective value f. j is obtained from i by selecting the best *move* (the operation that changes one solution to another). Set $i=j$.
Step 3: If $f(i)<f(i^*)$, then set $i^*=i$.
Step 4: Update the tabu and aspiration conditions.
Step 5: If a stopping condition is met, then stop. Else go to step 2.

In our algorithm, we start our heuristic by finding the MST. This gives us a lower bound (LB). The modified Kruskal algorithm provides us with a Degree Constrained Spanning Tree (DCST), which provides an upper bound (UB). The heuristic starts from the upper bound, which is feasible and work towards optimality. The moves are the set of edges incident with the leaves (vertices of degree 1) in the $G \backslash T$. Tabu tenure is set to be $0.1\ n$, where n is the number of vertices in the graph. The maximum number of iterations is $0.2n$. The stopping criteria are maximum number of iterations and the *tolerance*, where *tolerance* = 10 % of *gap* and *gap* = UB – LB. Note UB is revised as better feasible solutions are obtained.

The aspiration criteria are applied if a degree violation is detected. All possible edge exchanges among the edges of T incident to the violated vertex i and the edges of G not in T involving the neighbor of i, are examined. If searching doesn't yield any better solution, we record the current

best solution, put the currently used moves into tabu status and restart. The main idea of the algorithm is as follows:

```
begin
        t ← 0
   initialize: Graph, Tabu_move,Tabu_Status
    find MST, DCST
    control T ← DCST
 while (not termination condition)
     do
         choose the best move
         perform edge exchange
         if ( new solution feasible)
         compare the new solution with control T
         adjust the Tabu-move and tabu_status
          t ← t + 1
         else
         apply the aspiration criteria
         compare the new solution with control T
         adjust the Tabu-move and tabu_status
          t ← t + 1
    endif
  end
end
```

The Tabu Search method has not previously been applied to the DCMST problem.

4.2. Modified Penalty Algorithm

Our Modified Penalty (MP) algorithm is a slight variation of the Iterative Refinement (IR) method developed by Boldon et al. [1996] and Deo and Kumar [1997]. The IR method utilizes a special function called "Blacklisting function" to penalize the edges incident to the vertices violating the degree restriction. The penalty for edge (i,j) with $d_i > b_i$ and weight w_{ij} is determined as:

$$kt\left(\frac{w_{ij} - e_{\min}}{e_{\max} - e_{\min}}\right)e_{\max} \quad ,$$

where $0 \le k \le 1$, t =1 (if $d_j \le b_j$) or 2 (if $d_j > b_j$), and $e_{\max}$ and $e_{\min}$ are respectively the maximum and minimum edge weights in the current tree. They implemented the heuristics using parallel computing on massively parallel SIMD machine, MasPar MP-1 with 8192 processors.

The construction of the algorithms starts by finding a minimum spanning tree of the given graph. Then, the requirement (the degree restriction constraint) is used to increase the weights of selected edges in order that the next tree constructed will have fewer violations. After that, the minimum spanning tree of that graph is computed again using the edges that already have altered weights. This procedure is repeated until we get a spanning tree without degree violations.

The use of a 'blacklisting function' in this method is similar in spirit to the Tabu search. The two methods differ only in the search direction. Tabu search disallows search towards certain solutions in the combinatorial search space while in the Iterative Refinement, the blacklisting function guides the search by discouraging certain search directions.

For the sake of a quality comparison, we developed a code for the Iterative Refinement method. Since in the Iterative Refinement method, the smallest edge weight incidence to the violating vertex is kept for connectivity reason, we ensure that in the code, the smallest edge incident to every violated vertex is not penalized

We develop two versions, which are the modification of the Iterative Refinement method. The modifications we make are in the penalizing step and in the number of penalizing edges. In our method, every violated vertex is treated sequentially in every iteration. For the first version (MP1) we keep the smallest edge incident to every violated vertex not to be penalized and in the other version (MP2) we keep $b_i - 1$ smallest edges.

5. COMPUTATIONAL RESULTS.

We implemented our heuristics on a Silicon graphics Indy Machine with 150 MHz speed and 64 Mbytes memory. The tolerance is set 10% of the lower bound, and the maximum number of iterations is approximately 20% of the number of vertices. The time recorded is the time for finding the solutions, and does not include the system initialization.

We provide results on 2400 random problems generated as follows:

- Number of vertices range from 10 to 500.
- The edge weights are generated randomly from uniform distribution from 1 to 1000.
- For each n, 30 random problems are generated.
- For each n, graphs are generated with different v density p. We use p =0.25, 0.5, 0.75 and 1.0. For a given p, the edge e_{ij} is chosen if the random number q chosen from the unit distribution is less than p. The expected number of edges in the graph is $\binom{n}{2}p$. Disconnected graphs are rejected.

For Iterative Refinement and Modified Penalty methods, we implement the algorithms using two values of k, namely 0.5 and 1.

Some of the results are presented graphically in figures 1-8 below; we do not include all 2400 random problems in the figures. In all cases we use the degree bound of 3.

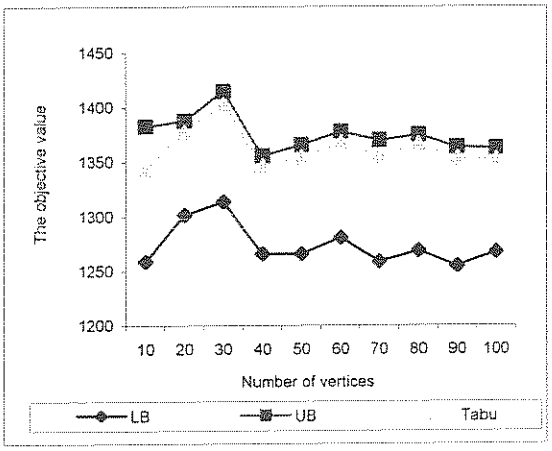

Figure 1. p=1

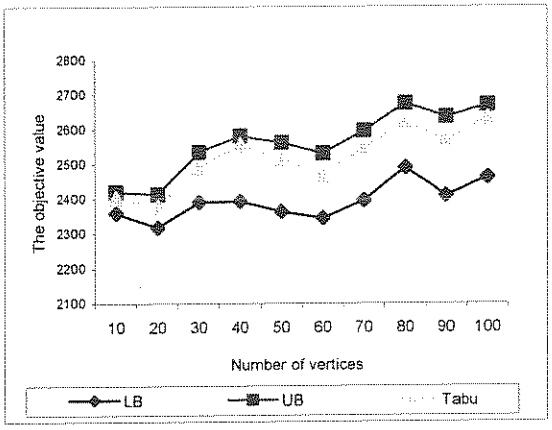

Figure 2. p=0.5

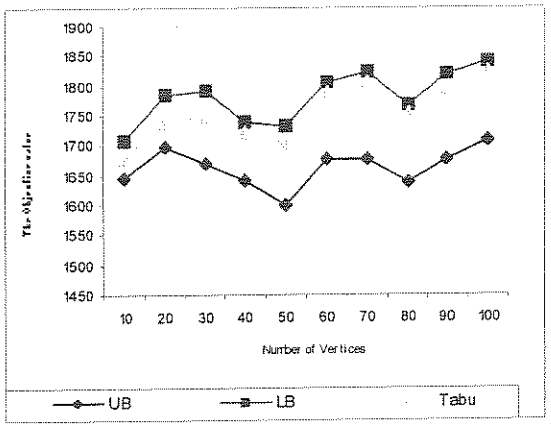

Figure 3. p=0.75

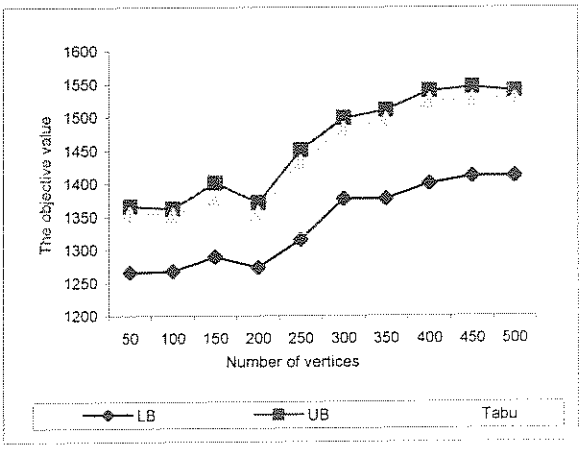

Figure 4. p=1

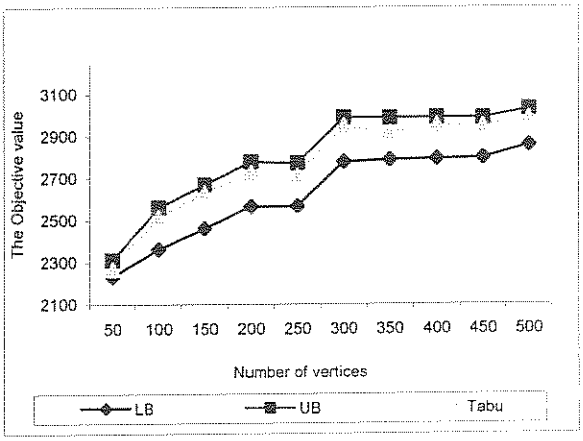

Figure 5. p=0.5

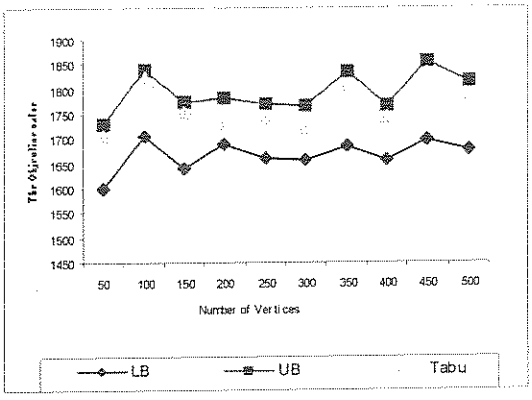

Figure 6. p=0.75

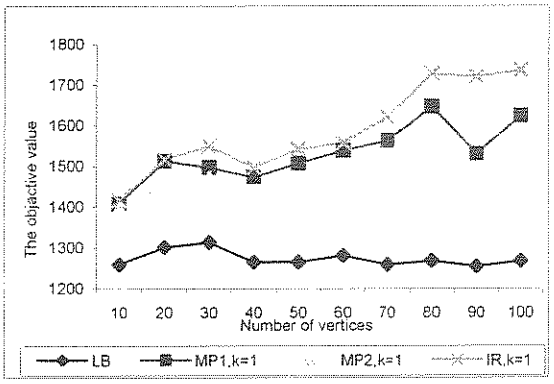

Figure 7. MP1, MP2 and IR with p=1 and k=1

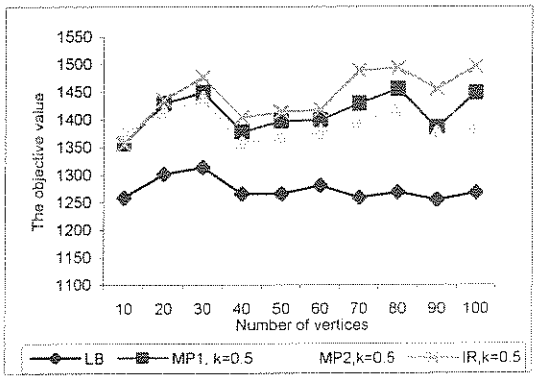

Figure 8. MP1, MP2, and IR with p=1, k=0.5

6. CONCLUSIONS

This paper developed a new approach based on Tabu search, to solve the DCMST problem. In addition, we presented a modified penalty method. Extensive computational results demonstrate that our Tabu search heuristic indeed improves the quality of the solution achieved by the upper bound. In the penalty approach, MP2 with k=0.5 is the best, followed by MP1, k=0.5. The performance of MP2, k=1 and IR, k=0.5 slightly the same, but when n increases, MP2, k=1 tends to perform better than IR, k=0.5

REFERENCES

Boldon, B., N. Deo and N. Kumar, Minimum Weight degree-constrained spanning tree problem: Heuristics and Implementation on an SIMD parallel machine. *Parallel Computing*, 22, 369–382, 1996.

Caccetta, L., B.K. Lam and S.P. Hill, Heuristics For the Degree restricted Spanning Tree Problem. Submitted for publication, 2000.

Caccetta, L. and S.P. Hill, A Branch and Cut Method for the Degree Constrained Minimum Spanning Tree Problem, *Networks, 37*, 74-83, 2001.

Deo N. and N. Kumar, Computation of Constrained Spanning Trees: A Unified Approach. *Network Optimization* (Lecture Notes in Economics and Mathematical Systems, Editor: Panos M. Pardalos, et al.). Springer-Verlag, Berlin, Germany, 194–220, 1997.

Garey, M.R., and D.S. Johnson, *Computers and Intractibility, A Guide to the Theory of NP-Completeness*. Freemann, San Francisco, 1979.

Gavish, B., Topological Design of Centralized Computer Networks Formulations and Algorithms. *Networks* 12, 355–377, 1982.

Glover Fred, and M. Laguna, *Tabu Search*. Kluwer Academic Publishers. Boston-Massachusetts, USA, 1997

Hertz, A., E. Taillard, and D. de Werra, Tabu Search. *Local Search In Combinatorial Optimization*, (Editor E. Aarts and J.K. Lenstra). John Wiley & Sons Ltd, New York, 121–136, 1997.

Krishnamoorthy, M., A.T.Ernst, and Y.M. Sharaiha, Algorithms for The Degree-Constrained Minimum Spanning Tree. Proceeding of Fourth International Conference on Optimization Techniques and Application. Perth, Western Australia, 1-3 July 1998, 859–866, 1998.

Krishnamoorthy, M., A.T.Ernst, and Y.M. Sharaiha, Comparison of Algorithms for the Degree Constrained Minimum Spanning Tree. Submitted to *Journal of Heuristics*, 2000.

Kruskal, J.B., On the Shortest Spanning Tree of a Graph and the Travelling Salesman Problem. *Proc.Amer.Math.Soc.* 7, 48-50, 1956.

Narula, S.C., and C.A. Ho, Degree-Constrained Minimum Spanning Tree. *Computer and Operation Research*, 7, 239-249, 1980.

Osman H. Ibrahim, and G. Laporte, Metaheuristics: A bibliography *Annals of Operations Research* 63, 513–623, 1996.

Prim, R.C., Shortest Connection Networks and Some Generalizations. *Bell System Technical Journal*, 36, 1389-1401, 1957.

Prufer, H., Neuer beweis eines satzes über Permutationen. *Arch.Math.Phys*, 27, 742-744, 1918.

Savelsbergh, M., and T.Volgenant, Edge Exchange In The Degree-Constrained Minimum Spanning Tree. *Computer and Operation Research* 12, 341-348, 1985.

Volgenant, A., A Lagrangean Approach to The Degree-Constrained Minimum Spanning Tree Problem. *European Journal Of Operational Research*, 39, 325-331, 1989.

Zhou, G., and M. Gen, A Note on Genetic Algorithms for Degree Constrained Spanning Tree Problems. *Network* 30, .91–95, 1997.