SOLVING THE DEGREE CONSTRAINED MINIMUM SPANNING TREE PROBLEM USING TABU AND MODIFIED PENALTY SEARCH METHODS

Wamiliana
Department of Mathematics, Faculty of Mathematics and Natural Sciences, Lampung University
E-mail: wamiliana@mail.com

ABSTRACT

In this paper we consider the Degree Constrained Minimum Spanning Tree Problem. This 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 center. Because of its NP-completeness, a number of heuristics have been proposed. In this paper we propose two new search methods: one based on the method of Tabu search and the other based on a penalty function approach. For comparative analysis, we test our methods on some benchmark problems. The computational results support our methods.

Keywords: minimum spanning tree, tabu search, degree constrained.

1. INTRODUCTION

The Degree Constrained Minimum Spanning Tree (DCMST) problem is concerned with finding a minimum-weight spanning tree whilst satisfying degree requirements on the vertices. The applications of the Degree Constrained Minimum Spanning Tree problems that may arise in real-life include: the design of telecommunication, transportation, and energy networks. It is also used as a subproblem in the design of networks for computer communication, transportation, sewage and plumbing. Gavish (1985), for example, used the DCMST as a subproblem in the design of a centralized computer network; and Gavish (1992) also provides several examples of optimization problems that are faced in the process of designing computer communication networks

The DCMST may be used in the design of the road system, which has to serve a collection of suburbs/towns, and has the additional restriction that no more than certain number of roads (example: four roads) are allowed to meet at an intersection. A degree constraint in a communication network also limits the liability in the case of vertex failure. In computer networks, the degree restrictions can be used to cater for the number of line interfaces available at a server/terminal.

The DCMST can be stated as follow: Given a weighted graph $G = (V,E)$ and positiveintegers $b_1, b_2, \ldots, b_n$, find a minimum weight spanning tree T such that the degree d_i of every vertex in T is at most $b_i, 1 \leq i \leq n$.

By reducing the DCMST problem to an equivalent symmetric Traveling Salesman Problem (TSP), Garey and Jhonson (1979) showed that this problem computationally difficult (NP-complete), thus it is unlikely that a polynomial bounded algorithm exists for solving general DCMST problems.

Many authors have proposed solution methods for the DCMST problem, which include both exact and heuristic methods. Since this problem is NP complete, heuristic methods have dominated. Some of the heuristics that have been investigated include: a number of basic MST algorithms of Prim (1957) and Kruskal (1956) (see Narula and Ho (1980) and Caccetta et al (2000)); the Genetic Algorithm by Zhou and Gen (1997); Simulated Annealing by Krishnamoorthy et al. (2001); Iterative Refinement by Boldon et al. (1996) and Deo and Kumar (1997); Tabu Search (CW1) by Caccetta and Wamiliana (2001).

Narula and Ho (1980) proposed two heuristics (primal and dual heuristics) to construct the DCMST. The primal heuristic is a modified Prim's algorithm and the other is the heuristic that starts with a minimum weight spanning tree generated from Prim's algorithm and moves towards feasibility through a series of edge exchanges. Their algorithms were tested on 120 randomly Euclidean problem problems with up to 100 vertices and degree bounds of 3 and 4. They do not implement with the more difficult problem, the random table problem.

Caccetta, et al. (2000) considered seven heuristics. The first, second and third heuristics are the modification of the Prim's and Kruskal's algorithms with the additional step of testing the degree violations. The fourth and fifth heuristics are also the modification of the Kruskal's algorithm with the introduction of a Critical Value (CV), where the $CV = a\left(\frac{LB}{n-1}\right)$. LB is the lower bound and the value of α used in calculations is 0.60. The CV is used for selecting edges where an edge is selected if its weight is less than the CV. The sixth heuristic is the heuristic of Savelsbergh and Volgenant (1985), and the seventh heuristic is the modification of the sixth heuristic by starting from one violated vertex as a root and using the procedure of Volgenant and Jonker (1983) in determining the cost of exchanging edges. They implemented these heuristics for the graphs with up to 500 vertices and degree bound of 3, 4, 5 and 6.

Boldon et al. (1996) and Deo and Kumar (1997) proposed an Iterative Refinement Method. They proposed 4 version of that method, named as algorithms BF1, BF2, BF3 and BF4. All algorithms start with finding a MST and then the edges incident to a degree violated vertex are penalized. The difference among the four lies on the number of penalizing edges and the penalty function. The penalty function for BF 1 and BF2 are deterministic (they name them as DET algorithms), while BF3 and BF4 are randomized (RAND algorithms). In addition, BF1 and BF3 penalized all edges incidence to the violated vertices, while BF2 and BF4 penalized all except the smallest one. With the new weighted edges, the process of calculating a MST is repeated, and continues until a spanning tree without degree violation is found. These algorithms were implemented 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). Among the four algorithms, BF2 performs the best. Problems with up to 5934 vertices were solved. However, the quality of the solution of this approach is not so good. They reported that for solving the DCMST with degree bound 5 their methods obtain the solution that lies within 40% to 70% of the MST (lower bound).

The Simulated Annealing method has been applied to the DCMST problem by Krishnamoorthy et al. (2001). In their Simulated Annealing procedure, they also used the Prufer (1918) number and a permutation of the nodes in the representation of solutions. The search strategy used is a *cut and paste* neighborhood transformation. This process starts by choosing randomly a vertex i from the spanning tree and then deleting the edge from i to j. This deletion of edge (i,j) creates two disconnected components. In their paper they did not mention which edge should be chosen if there are some edges incidence to vertex i. The two disconnected components are then re-connected by randomly choosing another node to connect i. To do this cut and paste method, they employed the permutation of the nodes, besides the representation of the tree by the Prufer number. They did this in order to avoid having to decode a number in each iteration.

The use of the Genetic algorithms for the DCMST problem has been investigated by Zhou and Gen (1997). They solved problems with up to 50 vertices. The Prufer (1918) number is employed in encoding and decoding the tree. In the method Zhou and Gen (1997) adopt uniform crossover and perturbation mutation operators as the genetic operators. The crossover probability used is 0.5 and the mutation probability is 0.01. This approach was tested on five randomly generated graphs with the number of vertices ranging from 10 to 50. The edge weights are integer values generated randomly from uniform distribution from 10 to 100.

Caccetta and Wamiliana (2001) and Wamiliana (2002), proposed Modified Penalty Methods (MP1 and MP2) as variants of Iterative Refinement methods. Implemented on some benchmark problems, the methods perform better than Simulated Annealing method.

The remainder of this paper is organized as follows. Section 2 discusses about the two heuristics approach for the DCMST problem, Section 3 presents the computational results and Section 4 contains conclusions.

2. THE TWO SEARCH METHODS PROPOSED

2.1 Tabu Search Approach

Tabu Search begins the same way as ordinary local or neighborhood search. It proceeds iteratively from one solution to another until a chosen termination criterion is satisfied. Each solution is reached from u by an operation called **move.** We note $w=u\oplus m \in S^*$ for the move applied to solution i in order to obtain solution w.

Tabu Search goes beyond local search by employing a strategy of modifying $N(u)$, the neighborhood of u, of solutions that are "close" to u as the search progresses, effectively replacing it by another neighborhood $N^*(u)$, using a special memory structure which serves to determine $N^*(u)$ *at every iteration,* and hence to organize the way in which the space is explored. Since the structure of the neighborhood depends on the iteration, we use notation $N(u,k)$ instead of $N^*(u)$.

The definition of $N(u,k)$ implies that some recently visited solutions are removed from $N(u,k\text{-}1)$. These solutions are considered as tabu solutions, which should be avoided in the next iteration. Such a memory based on recency will partially prevent cycling. For example, if L_k is a tabu list at iteration k, then the cycles of length at most $|L_k|$ will be avoided.

In order to improve the efficiency of the exploration process, one needs to keep track of not only local information (like current objective function value) but also some information related to the exploration process. Short-term memory keeps track of solution attributes that have changed during the past, and this memory is referred to as recency-based memory in Glover and Laguna (1997). Recency based memory is exploited by assigning a tabu active designation to selected attributes that occur in the solutions recently visited, and then the solutions that contain tabu active elements become tabu. However, these solutions only hold tabu status or tabu attributes temporarily, not forever. The tabu tenure is defined as the duration that an attribute remains tabu active (usually is measured by the number of iterations). Tabu tenure can vary for different types or combinations of attributes, can also vary over different times or stages of the search.

One term that accompanies the short-term memory is Aspiration Criteria or aspiration condition. The tabu status of a solution is not absolute, but can be overridden if certain conditions are met, and this is expressed in the form of aspiration condition. In effect, these aspiration conditions provide thresholds of attractiveness that the solutions maybe considered admissible in spite of being classified as tabu. Intensification strategy is another important component in long-term memory. Intensification 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.

Diversification strategy, as its names 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 solution that are infrequently used or alternatively, it may introduce such attributes by partially or fully restarting the solution process.

Adapting Tabu Search to the DCMST Problem.

In our heuristic we adapt the following terminology: *G* is an edge weighted connected graph, *T* is the minimum weight-spanning tree generated from, *Tabu move* is the set of edges that are given the status tabu active and kept on that attribute for certain time (tabu_tenure), *Tabu tenure* is the number of iterations that keep a solution in tabu status and *Moves* are the set of edges incidence to the leaves in *G\T*.

In our algorithm (called as CW), 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\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 0.2*n*. The stopping criteria are maximum number of iterations and the *tolerance*, where *tolerance*=1% 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
    set $t$: 0
    initialize: Graph, Tabu_move,Tabu_Status
    find MST, DCST
    set 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
        set $t$: $t + 1$
        else
        apply the  aspiration criteria
        compare the new solution with control T
        adjust the Tabu-move and tabu_status
        set $t$: $t + 1$
        endif
    end
end
```

2.2 Modified Penalty Approach

This method is the refinement of the previous version of the Modified Penalty algorithms (MP1 and MP2) developed by Caccetta and Wamiliana (2001), and Wamiliana (2002). In general, the Modified Penalty algorithm is the algorithms developed using the idea of the Iterative refinement method (IR) method of Boldon et al (1996) and Deo and Kumar (1997). The IR method starts by finding a minimum spanning tree of the given graph. Then, the degree restriction constraint is used to increase the weights of selected edges in order that the next tree constructed will have fewer violations. This step employs a special function called as “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:

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

where $0 \le k \le 1$, k is the *combing* factor, 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. Next, the minimum spanning tree of that graph is computed again using the edges that already have altered weights. Continue with this manner until a spanning tree without degree violations is found.

The penalizing (blacklisting) step has the effect of moving those edges that cause a degree violation low down in the sequence. This step is similar in ‘spirit’ with the Tabu Search method, where the penalized edges are ‘blacklisted’, while in the Tabu Search, the current moves are given tabu attributes/status. The new sequence of edges after resorting will determine the edges that appear in an MST in the next iteration. Thus, in the algorithm, the penalizing step is followed by the computation of the minimum spanning tree again.

We note here, that the quality of the final solution lies crucially on the 'blacklisting function', the function that penalized the edges incident to the violated vertices in the spanning tree. If the weights are increased by a large amount then the quality of the solution maybe not be so good, and too far from optimal. On the other hand, we face a slow convergence if the weights are increased by a small amount. The weight increased is controlled by a parameter in the blacklisting function, namely the combing factor k, where k is the real number whose value is a user setting and $0 \le k \le 1$.

The way Boldon et al. (1966) and Deo and Kumar (1997) defined the blacklisting functions motivates the modification we propose. The main modifications made are as follow:

1. b_i–1 smallest weighted edges incidence to the violated vertex i are not penalized.
2. The penalizing step is done sequentially in the loop routine, and there is no priority of handling the violated vertices. The violated vertices are handled one by one using the smallest index. Notice here that MP1 is quite similar to BF2, except in the implementation. In BF2 the penalizing step is handled simultaneously while in MP1 it is sequentially.

The main idea of the algorithm is as follow:

```
    Input: graph $G$, constraints C and D, where C is the minimum weight constraints and
    D is the degree constraints.
    begin
        In graph $G$ find a spanning tree that satisfies C
        while (spanning tree violates D)
            using D alter the weight of edges in $G$ by penalty function to obtain
            graph $G'$ with new weights
            In graph $G'$ find a spanning tree that satisfies C
            Set G: = $G'$
    end while
end
```

3. COMPUTATIONAL RESULTS

We implemented our heuristics on a Silicon graphics Indy Machine with 150 MHz speed and 64 Mbytes memory. We use 2160 random problems generated as follows:

- Number of vertices range from 10 to 500.
- The edge weights are integers and generated randomly from the uniform distribution (1, 1000).
- For each n, 30 random problems are generated
- For each n, graphs are generated with different 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. In all cases we use the degree bound of 3 which is computationally the hardest case.

For Tabu Heuristic approach, the tolerance is set 1% of the lower bound, and the maximum number of iterations is approximately 20 % of the number of vertices. For the Modified Penalty, we test the problems with size varies from 10 vertices to 250 vertices with the different increments: for vertex orders 10 to 100 in increments of 10, and in increments of 50 for higher orders. We test bigger sizes problems, but due to time limitations, some of the problems remained infeasible after the maximum iteration number (the allowable maximum iterations number is 10000) achieved. Based on our preliminary testing, using k value smaller than 0.5 will end up with slow convergence. In addition, testing with k =1 the performance is far from optimal.

Our computational results, based on those 2160 random table problems with up to 500 vertices and degree bound 3, the performance of the Tabu heuristic, in terms of the statistic $\frac{H-LB}{LB}$, reveal the average 6.51 % of the lower bound. We have to note here that in the implementation, especially for sparse graphs, the maximum iteration number must be set to a smaller value than the one already defined. This is because in sparse graphs there are not as many available moves as in the complete graphs.

The performances of the Modified Penalty (MP) algorithms strongly depend on the penalty function (Blacklisting function). Among the three components of the penalty function, the combing factor plays an important role. When k=1, the solution obtained on average is quite far from optimal, the average performance, in term of statistic $\frac{H-LB}{LB}$ is within 16.2242 %, while for k=0.5, the performances of the algorithm is within 11%.

For comparative analysis, we test our heuristics on some benchmark problems that used by Krisnamoorthy et al (2001) and Deo and Kumar (1997). Those problems can be downloaded from TSPLIB at: http://www.iwr.uniheidelberg.de/iwr/comopt/soft/ TSPLIB95/TSPLIB. The table below presents the computational results.

Table 1. Results on Some Benchmarks Problems

Algorithm	pr264	att532	rat575
Best TSP solution	49135	27686 (*)	6773
MST	41142	75872	6246
BF2(Deo, Kumar)	41143	NA	6265
BF4(Deo, Kumar)	41143	NA	6265
GA-F(Krishnamoorthy et al)	41142	75981	6250
GA-P (Krishnamoorthy et al)	44344	75981	6397
PSS(Krishnamoorthy et al)	41143	75981	6250
SA(Krishnamoorthy et al)	43438	79046	6393
BB(Krishnamoorthy et al)	41143	75912	6250
BB(Volgenant)	41143	75912	6250
Tabu (CW)	41154	75968	6265
Modified Penalty	41147	76091	6271

Legend (*): The att532 TSP result uses integer pseudo-Euclidean distances, so edge distances differ approximately by a factor $\sqrt{10}$.

4. CONCLUSION

This paper developed a new approach, based on Tabu search, to solve the Degree Constrained Minimum Spanning Tree Problem. Extensive computational results demonstrate that our Tabu search heuristic performs well. Moreover, tested on some benchmark problems, both methods, either Tabu search heuristic or Modified Penalty show that our heuristic is competitive.

REFERENCES

Boldon, B., N. Deo, N. Kumar, 1996. "Minimum Weight Degree-constrained Spanning Tree Problem: Heuristics and Implementation on an SIMD Parallel Machine", *Parallel Computing*, vol. 22, pp. 369 –382.

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

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

Caccetta, L., Wamiliana, 2001. "Heuristics Approach For The Degree Constrained Minimum Spanning Tree", *Proceeding of The International Modeling and Simulations*, pp. 2161-2166, Canberra.

Chris Tofides, N., 1976. *Worst-case Analysis of a New Heuristic for the Traveling Salesman Problem*, Carnegy-Mellon University, Pittsburgh, USA.

Deo N., N. Kumar, 1997. "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, pp. 194–220.

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

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

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

Held, M., R.M. Karp, 1970. "The Traveling Salesman Problem and Minimum Spanning Trees", *Operation Research*, vol. 18, pp.1138-1162.

Held, M., R.M. Karp, 1971. "The traveling Salesman Problem and Minimum Spaning Trees. Part II", *Mathematical Programming*, vol. 1, pp.6-25.

Krishnamoorthy, M., A.T. Ernst, Y.M. Sharaiha, 2001. "Comparison of Algorithms for the Degree Constrained Minimum Spanning Tree", *Journal of Heuristics*, vol. 7 no. 6, pp 587 – 611.

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

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

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

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

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

Volgenant A., R. Jonker, 1982. "A branch and Bound Algorithm for the Traveling Salesman Problem Based on the 1–tree Relaxation", *European Journal of Operational Research*, vol. 9, pp. 83-89.

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

Wamiliana, 2002. "The Modified Penalty Methods for The Degree Constrained Minimum Spanning Tree Problem", *Jurnal Sains dan Teknologi*, vol. 8, pp.1-12.

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