Variable Neighborhood Search for the Degree-Constrained Minimum Spanning Tree Problem

Celso C. Ribeiro

Department of Computer Science, Catholic University of Rio de Janeiro, Rio de Janeiro, RJ 22453-900, Brazil

Maurício C. Souza

Laboratoire d'Informatique, de Modélisation et d'Optimisation de Systèmes, Université Blaise Pascal, ISIMA, BP 125, 63173 Aubière Cedex, France

Abstract

Given an undirected graph with weights associated with its edges, the degree-constrained minimum spanning tree problem consists in finding a minimum spanning tree of the given graph, subject to constraints on node degrees. We propose a variable neighborhood search heuristic for the degree-constrained minimum spanning tree problem, based on a dynamic neighborhood model and using a variable neighborhood descent iterative improvement algorithm for local search. Computational experiments illustrating the effectiveness of the approach on benchmark problems are reported.

Key words: Combinatorial optimization, degree-constrained minimum spanning tree, local search, metaheuristics, variable neighborhood search.

1 Introduction

Let $G = (V, E)$ be a connected undirected graph, where V is the set of nodes and E denotes the set of edges. Given a non-negative weight function $w : E \to \mathbb{R}_+$ associated with its edges and a non-negative integer valued

Email addresses: `celso@inf.puc-rio.br` (Celso C. Ribeiro), `mauricio@isima.fr` (Maurício C. Souza).

[1] Work of this author was sponsored by the Brazilian National Research Council in the framework of doctorate scholarship number 200133/98-05.

degree function $b : V \rightarrow \mathbb{N}$ associated with its nodes, the Degree-Constrained Minimum Spanning Tree (DCMST) problem consists in finding a minimum spanning tree of G in terms of the edge weights w, such that the number of edges incident to each vertex $i \in V$ is less than or equal to b_i. The decision version of the DCMST problem is NP-complete (6), as far as it reduces to the shortest Hamiltonian path problem if $b_i = 2$ for all $i \in V$.

Narula and Ho (13) point out an application of this problem in the design of electrical circuits, in which some terminal nodes should be connected using a minimum amount of wire, with no more than a given number of wires incident to each terminal. Savelsbergh and Volgenant (14) mention two other applications: first, the design of a road system in which at most four roads are allowed to meet at any crossing; second, in communication networks, where a degree constraint limits vulnerability in case of drop out of a crossing. Gavish (7) reports an application in the design of computer networks.

Narula and Ho (13) propose primal and dual heuristic procedures and a branch-and-bound algorithm. Savelsbergh and Volgenant (14) present a branch-and-bound algorithm which makes use of an edge elimination procedure based on edge exchange considerations. Edge exchanges are also used in a branch-and-bound algorithm based on Lagrangean relaxation proposed by Volgenant (16). Lagrangean relaxation was also applied by Gavish (7) to obtain lower bounds for the degree-constrained minimum spanning tree problem. Implementations of metaheuristics are among the most efficient approximate algorithms to find an optimum DCMST. Zhou and Gen (17; 18) developed a genetic algorithm that uses the Prüfer number as a tree encoding. Craig et al. (3) and Krishnamoorthy et al. (11) proposed several heuristics including simulated annealing, genetic algorithms, and problem space search, as well as an enumeration-based exact solution approach.

We propose a Variable Neighborhood Search (VNS) heuristic for the DCMST problem, using a Variable Neighborhood Descent (VND) strategy for local search. The VNS heuristic and the associated dynamic neighborhood model are presented in details in the next section. In Section 3 we describe the hierarchy of neighborhood structures used at each local search phase. Local search within each VNS iteration is performed by a VND iterative improvement algorithm presented in Section 4. Computational results are reported in Section 5. Concluding remarks are presented in the last section.

2 VNS Heuristic

The Variable Neighborhood Search (VNS) metaheuristic proposed by Hansen and Mladenović (10; 12) is based on the exploration of a dynamic neighborhood model. Given any feasible solution $T = (V, A)$ to the DCMST problem, with $A \subseteq E$ and $|A| = |V| - 1$, we denote its cost by $w(T) = \sum_{e \in A} w(e)$. Let $N_{vns}^{(k)}(T)$ be the set of all feasible trees T' which have at least $|V| - 1 - k$ edges in common with T (i.e., there are at most k edges which appear either in T or in T', but not in the other). Sequence $N_{vns}^{(1)}, \ldots, N_{vns}^{(k_{\max})}$, with $k_{max} \leq |V| - 1$, defines a neighborhood structure for the DCMST problem, so that solutions in higher order neighborhoods of the same solution T are increasingly different from T.

VNS successively explores increasing order neighborhoods in the search for improving solutions. A current solution T and a neighborhood order k are associated with each VNS iteration. Each iteration has two main steps. In the first step, a neighbor solution T' is randomly generated within neighborhood $N_{vns}^{(k)}$ of solution T. Next, local search is applied to T'. If the new solution $\bar{T}$ found by local search improves the current solution, the algorithm resumes the search from this solution using the first neighborhood. Otherwise, the algorithm resumes from T using a higher order neighborhood. The algorithm stops after performing *seqs* times the exploration of the complete sequence $N_{vns}^{(1)}(T), \ldots, N_{vns}^{k_{\max}}(T)$ of neighborhoods, without finding any improving solution. Figure 1 gives the algorithmic desciption of the procedure which implements the VNS metaheuristic for the DCMST problem.

procedure `VNS_DCMST`
    Let T_0 be an initial solution.
    $T \leftarrow T_0$
    Set $k_{\max}$ to some value in $\{2, \ldots, |V| - 1\}$.
    for $i = 1, \ldots, seqs$ **do**
        $k \leftarrow 1$
        while $k \leq k_{\max}$ **do**
            Randomly generate $T' \in N_{vns}^{(k)}(T)$
            Obtain $\bar{T}$ by applying a local search procedure to T'.
            if $w(\bar{T}) < w(T)$
            then $T \leftarrow \bar{T}$; $k \leftarrow 1$;
            else $k \leftarrow k + 1$;
            end-if
        end-while
    end-for
end-procedure

Fig. 1. VNS heuristic for the DCMST problem

Initial solutions are randomly constructed. We start with a forest formed by all isolated nodes. At each iteration, an edge is randomly selected. If its extremities are in different connected components of the solution under construction and if its insertion does not violate any of the two associated degree constraints, then this edge is inserted into the tree; otherwise it is discarded. Assuming that G is a complete graph and that $b_i \geq 2, \forall i \in V$, then the algorithm always stops and finds a spanning tree of G.

Neighbor solutions T' within $N_{vns}^{(k)}(T)$ are generated as follows. First, k edges of the current solution $T = (V, A)$ are randomly eliminated from A, breaking T into k connected components. A new tree T' is rebuilt by the insertion of k randomly selected edges, so as to put together the connected components and satisfy the degree constraints. For each edge, checking whether it links two different connected components while preserving feasibility can be done in $O(\log |V|)$ time using the union-finding algorithm (see e.g. Aho and Ullman (1)). Since in the worst case all edges may be checked for insertion, the generation of each neighbor within $N_{vns}^{(k)}(T)$ can be done in $O(|E| \log |V|)$ time. The next step consists in applying a local search strategy to T'. An iterative improvement algorithm for local search based on another dynamic neighborhood model is described in the forthcoming sections.

3 Neighborhood Structure for Local Search

Local search methods are based on the investigation of the solution space, by successively exploring the neighborhood of a current solution and criteriously moving to one of its neighbors whenever an improving move is found. Variable neighborhood methods for local search are based on exploring neighborhood changes. The search focus into small neighborhoods whenever possible, moving to higher order neighborhoods only when no further improvement is possible within the one under investigation.

As before, let $T = (V, A)$ be a spanning tree of graph G satisfying all degree constraints, with $A \subseteq E$ and $|A| = |V| - 1$. The edges not in T are divided into three sets

- E_0: edges whose both extremities are not saturated,
- E_1: edges with exactly one extremity saturated, and
- E_2: edges whose both extremities are saturated,

with $E = A \cup E_0 \cup E_1 \cup E_2$. An edge-exchange operation is defined by a pair (u, v) of edges, with $u \in A$ and $v \in E \setminus A$, such that the resulting graph $T' = (V, (A \cup \{v\}) \setminus \{u\})$ is still a spanning tree of G. We describe below three neighborhood relations $N_{vnd}^{(1)}$, $N_{vnd}^{(2)}$, and $N_{vnd}^{(3)}$ for the DCMST problem, based respectively on the application of one, two, or three edge-exchange operations.

Neighborhood $N_{vnd}^{(1)}$ is defined by all edge-exchange operations (u, v) which preserve feasibility. If edge $v \in E_0$, then the edge u to be removed is that with the maximum weight among those in the cycle induced by the insertion of v into T. If $v \in E_1$, then necessarily u is the edge in this cycle which is adjacent to the saturated extremity of v in T. The number of moves in $N_{vnd}^{(1)}$ is $O(|E_0| + |E_1|)$. In case $v \in E_1$, each move value $w_v - w_u$ may be evaluated in time proportional to the degree of the saturated extremity of v. In case $v \in E_0$, the maximum weight edge has to be determined, which can be done in time linear in the number of edges in the cycle.

Moves in neighborhood $N_{vnd}^{(2)}$ are defined by a sequence of two edge-exchange operations (u_1, v_1) and (u_2, v_2). The first exchange operation leads to a tree with exactly one vertex violating the degree constraints, while the second one restores feasibility. If edge $v_1 \in E_1$, then the edge u_1 to be removed is that with the maximum weight among those in the cycle induced by the insertion of v_1 into T. If $v_1 \in E_2$, then u_1 is the maximum weight edge among the two in the cycle which are adjacent to the extremities of v_1 (since both are saturated in T). Let $\hat{T}$ be the (infeasible) tree obtained after the first edge-exchange operation and let p be its unique vertex violating the degree constraints. The second exchange operation restores feasibility, by (i) removing an edge u_2 adjacent to p and (ii) inserting an edge $v_2 \in E_0 \cup E_1$ connecting the two components created in $\hat{T}$, respecting the degree constraints. All pairs (v_2, u_2) are considered as candidate moves. The number of moves in $N_{vnd}^{(2)}$ is $O((|E_1| + |E_2|) \times (|E_0| + |E_1|))$. In this neighborhood, the value of a move is given by the difference $w_{v_1} + w_{v_2} - w_{u_1} - w_{u_2}$ between the weights of inserted and removed edges. The complexity of the first exchange operation is defined by the determination in linear time of the edge to be removed from the cycle. The second exchange operation requires a depth-first search traversal of graph G. In consequence, the complexity of each move evaluation is $O(|V| + |E|)$.

Finally, moves in neighborhood $N_{vnd}^{(3)}$ are defined by a sequence of three steps associated with edge-exchange operations (u_1, v_1), (u_2, v_2), and (u_3, v_3). The first edge-exchange operation leads to a solution violating exactly two degree constraints. The second operation restablishes the feasibility of one of these constraints, while the last one leads to a feasible solution satisfying all degree constraints. In this case, $v_1 \in E_2$ and u_1 is the largest weight edge in the cycle induced in T by the insertion of v_1. Let $\hat{T}$ be the (infeasible) tree obtained after the first edge-exchange operation and let p and q be the two vertices whose degree constraints are violated. As for the previous neighborhood, the second edge-exchange operation restablishes the feasibility of one of these two vertices, by removing an edge u_2 adjacent to p in $\hat{T}$ and inserting a new edge $v_2 \in E_0 \cup E_1$ connecting the two components created in $\hat{T}$. Finally, the last edge-exchange operation restores feasibility by removing an edge u_3 adjacent to q in the current tree and inserting an edge $v_3 \in E_0 \cup E_1$ connecting the two components created in this tree. Neighborhood $N_{vnd}^{(3)}$ has $O(|E_2| \times |E_0 \cup E_1|^2)$ possible moves. However, in our implementation only $O(|E_2| \times |E_0 \cup E_1|)$ moves are effectively evaluated, since the third operation is evaluated exclusively for one selected pair involved in the chosen second operation. As for neighborhood $N_{vnd}^{(2)}$, the value of a move is given by the difference $w_{v_1} + w_{v_2} + w_{v_3} - w_{u_1} - w_{u_2} - w_{u_3}$

between the weights of inserted and removed edges. The complexity of the first exchange operation is defined by the determination in linear time of the edge to be removed from the cycle. The two remaining exchange operations require each a depth-first search traversal of graph G. In consequence, the complexity of each move evaluation within this neighborhood is $O(|V| + |E|)$.

We use the data structures proposed in (2; 8; 9) to speedup the computation of move values. To conclude this section, we note that this neighborhood hierarchy presents an increasing number of moves, although they are pairwise disjoint. Moves in the first neighborhood are faster to be evaluated than those in the two other neighborhoods.

4 VND Local Search

Let $N_{vnd}^{(1)}$, $N_{vnd}^{(2)}$, and $N_{vnd}^{(3)}$ be neighborhood structures for the DCMST problem, as defined in the previous section. Dynamic neighborhood methods for local search are based on exploring neighborhood changes, determined by specific events occured during the search. We investigate an iterative improvement procedure based on variable neighborhoods, which essentially is a variant of the Variable Neighborhood Descent (VND) metaheuristic proposed by Hansen and Mladenović (10; 12).

The VND local search strategy performs an iterative improvement procedure starting from the neighbor solution T' being explored by procedure `VNS_DCMST` presented in Section 2, using the first neighborhood $N_{vnd}^{(1)}$. Whenever a local optimum is found, the current neighborhood is changed. In case an improving move is found, the algorithm continues the search exploring again the first neighborhood. The algorithm stops when a local optimum is found within the last neighborhood. Figure 2 gives the algorithmic description of the procedure which implements this local search approach.

5 Computational Results

In this section, we present numerical results obtained with the application of the `VNS_DCMST` algorithm proposed in this work, using the iterative improvement `VND_DCMST` procedure for local search. The algorithm was implemented in `C`, using version 2.8.1 of the `gcc` compiler for AIX, with the optimization flag set to `-O`. The random number generator of Schrage is used (15). All computational experiments have been performed on a 200 MHz RISC 6000 workstation model 43P-140 with 64 Mbytes of RAM memory.

```
procedure VND_DCMST(T')
    ℓ ← 1
    while ℓ ≤ 3 do
        Find the best solution T'' ∈ N_vnd^(ℓ)(T').
        if w(T'') < w(T')
        then T' ← T''; ℓ ← 1;
        else ℓ ← ℓ + 1;
        end-if
    end-while
    T̄ ← T';
    return T̄
end-procedure
```

Fig. 2. VND iterative improvement algorithm for local search

The first set of test problems we considered in our computational experiments are five instances labelled ZG described by Zhou and Gen (17; 18). They are associated with complete graphs with 10, 20, 30, 40, and 50 vertices. Edge weights are randomly generated with a uniform distribution within $[10, 100]$. All maximum degrees are equal to three.

Krishnamoorthy et al. (11) conducted a computational study of heuristics for the DCMST problem on four different sets of randomly generated problems: CRD, SYM, STR, and SHRD. CRD and SYM are randomly generated Euclidean problems in complete graphs with up to 100 vertices, with node degrees in the constrained spanning tree bounded by three. For both CRD and SYM problem classes, ten instances have been generated for each graph size. These same problem types were also used in Volgenant (16). According with Krishnamoorthy et al. (11), problem classes STR and SHRD have been generated with the purpose of characterizing more difficult instances. In series STR, nodes are distributed in clusters within Euclidean spaces of given dimension. The space dimension varies from 3 to 7. For each value of $|V| = 30$, 50, and 70, the maximum degree ranges from 4 to 5. For $|V| = 100$, the maximum degree ranges from 3 to 5. Two instances have been generated for each combination of maximum degree and space dimension, in a total of 45 pairs of instances in this class. The 24 non-euclidean problems in class SHRD are such that there are two instances for each value of $|V| = 15$, 20, 25, and 30, and maximum degrees equal to 3, 4, and 5.

Tables 1-3 summarize the results obtained for the 24 instances of the more difficult series SHRD. For each parameter setting or algorithmic strategy, each table reports the average relative error (with respect to the best solution found along this study), the number of instances for which the best solution was found, and the average computation time in seconds. In the first two tables we study the influence of the two parameters characterizing the implementation of algorithm VNS_DCMST: the stopping criterion, defined by the number *seqs* of times the highest order neighborhood is explored without finding an improving solution, and the number $k_{\max}$ of edges removed to generate a solution in the highest order neighborhood $N_{vns}^{(k_{\max})}$. In Table 1, we set parameter $seqs = 1$ and report run statistics for $k_{\max}$ equal to 10%, 20%, 30%, 40%, and 100% of the number $|V| - 1$ of edges in any feasible solution.

Table 1
Effect of parameter $k_{\max} = \lceil \alpha\ (|V| - 1) \rceil$ ($seqs = 1$)

Instances SHRD	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 1.0$
average relative error (%)	0.26	0.23	0.03	0.00	0.00
best solutions found	11	14	20	23	24
average time (seconds)	0.04	0.09	0.15	0.21	0.59

Since no significant improvement in solution quality is found for $k_{\max}$ larger than 40% of the number of edges $|V| - 1$, we set $k_{\max} = \lceil 0.4\ (|V| - 1) \rceil$ and report in Table 2 overall results obtained for the same instances with parameter $seqs$ equal to 1, 2, 3, 4, and 5. Again, as no significant improvement in solution quality is found by investigating the full neigborhood sequence more than once, parameter $seqs$ is set to one.

Table 2
Effect of parameter $seqs$ ($k_{\max} = \lceil 0.4\ (|V| - 1) \rceil$)

Instances SHRD	$seqs = 1$	$seqs = 2$	$seqs = 3$	$seqs = 4$	$seqs = 5$
average relative error (%)	0.00	0.00	0.00	0.00	0.00
best solutions found	23	23	24	24	24
average time (seconds)	0.21	0.35	0.48	0.55	0.73

Five algorithmic strategies are evaluated and their results compared in Table 3: random construction of the initial solution (Start); local search applied to the initial solution using exclusively the first neighborhood $N_{vnd}^{(1)}$ (LS); VND iterative improvement for local search applied only to the initial solution and using the three neighborhoods (VND); and the complete `VNS_DCMST` algorithm with parameters $k_{\max} = \lceil 0.4\ (|V| - 1) \rceil$ and $seqs = 1$, using procedure `VND_DCMST` for local search (VNS). This table shows that the VND iterative improvement algorithm significantly improved the solutions obtained by a simple local search strategy exclusively based on the first neighborhood $N_{vnd}^{(1)}$, reducing the average relative error from 4.79% to 0.27% with a very small increase in computation time. VNS leads to still better results, finding best solutions for 23 out of the 24 instances of series SHRD.

We now compare the results obtained by the VNS heuristic using VND for local search with other approaches in the literature. We first compare algorithm `VNS_DCMST` with the genetic algorithm of Zhou and Gen (17; 18), using their same five test problems. Comparative results are presented in Table 4, in

Table 3
Comparative results for different algorithmic strategies

Instances SHRD	Start	LS	VND	VNS
average relative error (%)	228.42	4.79	0.27	0.00
best solutions found	0	0	9	23
average time (seconds)	0.00	0.00	0.04	0.21

which we report problem characteristics (instance identification, number of nodes, and maximum degree limit value), a lower bound given by the weight of the minimum spanning tree, and the solution values found by each algorithm (together with the relative deviation from the above lower bound). We also give computation times in seconds for VNS, which obtains much better solutions for four out of the five test problems in this set. For the remaining instance, both algorithms found the same solution. Computation times for the genetic algorithm were not available.

Table 4
Test problems: Zhou and Gen (17; 18)

Problem				GA		VNS		
Instance	$\|V\|$	b_i	LB	value	deviation (%)	value	deviation (%)	time
ZG1	10	3	117	123	5.13	123	5.13	0.00
ZG2	20	3	233	237	1.72	233	0.00	0.00
ZG3	30	3	316	327	4.48	317	0.32	0.03
ZG4	40	3	419	449	7.16	420	0.24	0.09
ZG5	50	3	513	554	7.99	514	0.19	0.22

Krishnamoorthy et al. (11) proposed three different heuristic approaches for the DCMST problem: another genetic algorithm (GA-F), problem space search (PSS), and simulated annealing. They conducted a computational study on a 200 MHz DEC Alpha processor, using four different sets of randomly generated problems: CRD, SYM, STR, and SHRD. The next phase of our computational experiments addresses the comparison of our VNS approach with the best results in (11).

For instances CRD and SYM, the problem space search (PSS) heuristic lead to the best results in (11). Average relative deviations from the optimum value and average computation times in seconds over the 10 instances within each series and for each problem size are reported in Table 5. VNS always obtained better solutions (i.e., smaller average deviations from the optimum value) than PSS in smaller computation times. For 69 out of the 70 instances in these classes we found the exact optimal solutions. For the remaining instance, algorithm `VNS_DCMST` found a solution only three units out of the optimal value

(1371 instead of 1368).

Table 5
Test problems: CRD and SYM

Problem			PSS		VNS	
Class	$\lvert V\rvert$	b_i	deviation (%)	time	deviation (%)	time
CRD	30	3	0.04	5.80	0.00	0.02
CRD	50	3	0.04	18.74	0.00	0.13
CRD	70	3	0.00	40.06	0.00	0.52
CRD	100	3	0.00	87.33	0.00	2.46
SYM	30	3	0.04	9.34	0.02	0.03
SYM	50	3	0.30	45.01	0.00	0.18
SYM	70	3	2.53	99.34	0.00	0.60

We now address the evaluation of results obtained for test problems SHRD, generated with the purpose of characterizing more difficult instances (11). For each problem in this series, we report in the first four columns of Table 6 the name of the corresponding file, the number of vertices, the maximum value associated with the degree constraints, and the best known solution found so far. The solution found by the complete VNS heuristic and the associated computation time in seconds are reported in the two last columns. These results show that algorithm `VNS_DCMST` was quite effective and improved the best known solutions for 20 out of the 24 instances. For the remaining four problems, VNS found same quality solutions. VNS lead to an average improvement of 1.02% with respect to the best known solutions for the 24 instances in this class. Computation times are quite small.

Finally, we report computational results for STR problems. For this series, the genetic algorithm GA-F lead to the best results in (11). Each row of Tables 7-8 reports average results for a pair of instances in this series with the same characteristics, as reported in (11) to allow for a fair comparison. The first three columns give their number of vertices, maximum value associated with the degree constraints, and space dimension. The following average results for each pair of instances are reported for both the genetic algorithm GA-F and our VNS heuristic: the average relative gap with respect to the optimal solution (best known solutions for instances marked with the symbol '*') and the average computation time in seconds. Optimal or currently best known solutions have been provided by Ernst and Krishnamoorthy (4). This class seems to be that for which VNS obtained more significant improvements with respect to previous heuristics. VNS found optimal or best known solutions for all instances in this class. It improved the average best solution values found by the GA-F for 35 out of the 45 pairs of instances. Moreover, for three instances among them (marked with the symbol '§') VNS found solutions even better than the best known solutions reported by Ernst and Krishnamoorthy (4). Once again, computation times are quite small and significantly better than those observed with the genetic algorithm.

Table 6
Test problems: SHRD

File	$\|V\|$	b_i	Best	VNS	time
SHRD150	15	3	582	582	0.03
SHRD150	15	4	433	430	0.02
SHRD150	15	5	339	339	0.01
SHRD159	15	3	598	597	0.01
SHRD159	15	4	430	430	0.02
SHRD159	15	5	332	332	0.02
SHRD200	20	3	1105	1088	0.12
SHRD200	20	4	818	802	0.06
SHRD200	20	5	632	627	0.09
SHRD209	20	3	1114	1092	0.04
SHRD209	20	4	808	799	0.06
SHRD209	20	5	630	629	0.03
SHRD258	25	3	1786	1745	0.25
SHRD258	25	4	1290	1277	0.30
SHRD258	25	5	1007	999	0.12
SHRD259	25	3	1792	1756	0.20
SHRD259	25	4	1309	1292	0.15
SHRD259	25	5	1025	1016	0.22
SHRD300	30	3	2645	2592	0.47
SHRD300	30	4	1937	1905	0.37
SHRD300	30	5	1526	1504	0.32
SHRD309	30	3	2621	2585	0.71
SHRD309	30	4	1911	1898	0.78
SHRD309	30	5	1486	1474	0.84

Table 7
Test problems: STR ($|V| = 30, 50$)

Problem			GA-F		VNS	
$\|V\|$	b_i	dimension	deviation (%)	time	deviation (%)	time
30	4	3	0.07	4.17	0.00	0.05
30	4	4	0.00	4.10	0.00	0.06
30	4	5	0.00	3.81	0.00	0.07
30	4	6	0.08	4.26	0.00	0.08
30	4	7	* 0.11	4.03	* 0.00	0.07
30	5	3	0.00	3.40	§ -0.01	0.04
30	5	4	0.12	3.35	0.00	0.06
30	5	5	0.00	3.33	0.00	0.05
30	5	6	0.09	3.45	0.00	0.07
30	5	7	* 0.17	3.46	* 0.00	0.07
50	4	3	0.00	24.34	0.00	0.32
50	4	4	0.00	23.04	0.00	0.39
50	4	5	0.72	24.11	0.00	0.46
50	4	6	0.15	22.57	0.00	0.51
50	4	7	* 0.57	23.66	* 0.00	0.54
50	5	3	0.03	19.87	0.00	0.24
50	5	4	0.00	19.60	0.00	0.35
50	5	5	0.54	20.00	0.00	0.40
50	5	6	0.07	19.48	0.00	0.49
50	5	7	* 0.32	19.70	* 0.00	0.51

6 Concluding Remarks

We presented a VNS heuristic for the DCMST problem. Local search is peformed by a VND iterative improvement algorithm. Computational results obtained for some benchmark problems have shown that the VNS heuristic is quite effective, finding better results than other heuristics within very small computation times. We summarize the results obtained for the benchmark instances in series STR and SHRD from (11). VNS found optimal or best known solutions for all problems in series STR, improving the best known solutions for instances in 35 out of the 45 pairs of instances in this series.

Table 8
Test problems: STR ($|V| = 70, 100$)

Problem			GA-F		VNS	
$\|V\|$	b_i	dimension	deviation (%)	time	deviation (%)	time
70	4	3	0.07	70.23	0.00	1.27
70	4	4	0.58	68.68	0.00	1.55
70	4	5	0.68	65.28	0.00	1.78
70	4	6	0.22	69.48	0.00	2.57
70	4	7	* 0.56	68.53	* 0.00	2.34
70	5	3	0.00	59.67	0.00	0.98
70	5	4	0.12	59.89	0.00	1.39
70	5	5	0.57	61.00	0.00	1.56
70	5	6	0.09	59.83	0.00	1.84
70	5	7	* 0.52	57.97	* 0.00	2.11
100	3	3	0.17	257.55	0.00	7.58
100	3	4	1.10	271.85	§ -0.01	7.39
100	3	5	0.07	314.29	0.00	7.09
100	3	6	0.58	203.85	§ -0.01	11.52
100	3	7	0.58	233.01	0.00	8.88
100	4	3	0.00	245.80	0.00	5.55
100	4	4	0.38	240.35	0.00	6.79
100	4	5	0.02	254.46	0.00	7.55
100	4	6	0.51	233.18	0.00	8.62
100	4	7	* 0.51	232.48	* 0.00	9.15
100	5	3	0.00	220.06	0.00	4.14
100	5	4	0.05	213.79	0.00	6.45
100	5	5	0.00	219.51	0.00	7.08
100	5	6	0.46	206.02	0.00	7.73
100	5	7	* 0.48	207.28	* 0.00	8.83

VNS also found better solutions than those currently known for 20 out of the 24 instances in the most difficult class SHRD of test problems. We have also established the superiority of the VND local search strategy with respect to a single neighborhood approach.

Some extensions of this approach are currently under development, including improved initial solution construction procedures and hybridization of the VND iterative improvement algorithm within the local search phase of a GRASP (5) for the DCMST problem. The VNS/VND approach is also being extended to the solution of the capacitated minimum spanning tree problem.

Acknowledgements: The authors are grateful to G. Zhou, M. Gen, M. Krishnamoorthy, and A. Ernst, for making available their test problems.

References

[1] A.V. Aho e J.D. Ullman, *Foundations of Computer Science, Vol. I*, W.H. Freeman, 1992.

[2] R. Barr, F. Glover, and D. Klingman, "Enhancements of spanning tree labelling procedures for network optimization", *INFOR* 17 (1979), 16–34.

[3] G. Craig, M. Krishnamoorthy, and M. Palaniswami, "Comparison of heuristic algorithms for the degree constrained spanning tree", in *Metaheuristics: Theory and Applications* (I.H. Osman and J.P. Kelly, eds.), Kluwer, 1996.

[4] A. Ernst and M. Krishnamoorthy, personnal communications, 1999/2000.

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

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

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

[8] F. Glover and D. Klingman, "Recent developments in computer implementation technology for network flow algorithms", *INFOR* 20 (1982), 433–452.

[9] F. Glover, D. Klingman, R. Krihnan, and R. Padman, "An indepth empirical investigation of non-greedy approaches for the minimum spanning tree problem", *European Journal of Operational Research* 56 (1992), 343–356.

[10] P. Hansen and N. Mladenović, "An introduction to variable neighbourhood search", in *Metaheuristics: Advances and Trends in Local Search Procedures for Optimization* (S. Voss, S. Martello, I.H. Osman, and C. Roucairol, eds.), 433–458, Kluwer, 1999.

[11] M. Krishnamoorthy, A.T. Ernst, and Y.M. Sharaiha, "Comparison of algorithms for the degree constrained spanning tree", submitted to *Journal of Heuristics*, 1997.

[12] N. Mladenović and P. Hansen, "Variable neighbourhood search", *Computers and Operations Research* 24 (1997), 1097–1100.
[13] S.C. Narula and C.A. Ho, "Degree-constrained minimum spanning tree", *Computers and Operations Research* 7 (1980), 239–249.
[14] M. Savelsbergh and T. Volgenant, "Edge exchanges in the degree-constrained minimum spanning tree problem", *Computers and Operations Research* 12 (1985), 341–348.
[15] L. Schrage, "A more portable Fortran random number generator", *ACM Transactions on Mathematical Software* 5 (1979), 132–138.
[16] T. Volgenant, "A Lagrangean approach to the DCMST problem", *European Journal of Operational Research* 39 (1989), 325–331.
[17] G. Zhou and M. Gen, "A note on genetic algorithms for the degree-constrained spanning tree problem", *Networks* 30 (1997), 91–95.
[18] G. Zhou and M. Gen, "Approach to degree-constrained minimum spanning tree problem using genetic algorithm", *Engineering Desing and Automation* 3 (1997), 157–165.