Reduction Tests for the Steiner Problem in Graphs

C. W. Duin and A. Volgenant
Department of Operations Research, Faculty of Economic Sciences and Econometrics, University of Amsterdam, Jodenbreestraat 23, 1011 NH, Amsterdam, The Netherlands

Before actually solving the Steiner problem in graphs, it is known that reduction tests may reduce the problem size, e.g., by eliminating vertices from the graph. We improve existing tests and develop new techniques based on a bottleneck approach. The latter include optimal edge detection, edge elimination, and node elimination because of degree considerations. We give computational results of a reduction algorithm on a large number of test problems. These problems have up to 200 vertices; their edge densities vary from sparse to complete with Euclidean weights as well as uniformly random. The algorithm solves the majority of the test problems by reduction tests only, and reduces the size of the remaining problems to at most one fourth of the original sizes.

1. INTRODUCTION

The Steiner problem in graphs (for short SPG) is derived from a problem in geometry known as Steiner's problem (SP). This is the problem of connecting a given set of points in the Euclidean plane by straight lines, in the sense that there is a path of lines between every pair in the set, so as to minimize the total length of the lines used. In general an optimal solution is not a (minimum) spanning tree on the given points, because the line segments may intersect anywhere in the plane. Exact algorithms due to Melzak [12] and Cockayne [4] can only solve small problems. Computationally seen the SPG is easier to solve, although NP-hard. It can be described as follows.

> Let $G = (V,K,E,c)$ denote an undirected graph with V the set of vertices, K a subset of V called the set of *special* vertices, E the set of undirected edges and c a positive function on E. The SPG requires the identification of a connected subgraph T that includes all the special vertices of K, the criterion being to minimize $c(T) = \Sigma_{(i,j)\in T}\, c_{ij}$.

(Edges with nonpositive costs are eliminated at forehand by fusing their end vertices.) An optimal solution is called a *Steiner tree*. It is a minimum-cost spanning tree on some set of vertices $K \cup S$; the non special vertices of S are called *Steiner vertices*.

NETWORKS, Vol. 19 (1989) 549–567

CCC 0028-3045/89/050549-19$04.00

For extensive surveys on the SPG, we refer to Winter [16] and Maculan [11]. Several solution techniques have been developed, heuristics—see e.g., Takahashi and Matsuyama [15] or more recently Rayward-Smith and Clare [13]—as well as exact algorithms. In the case $|E| = |V| - 1$, the given graph is a tree that can be pruned into a Steiner tree. For $|K| = 2$ the SPG reduces to a shortest path problem and for $|K| = |V|$ it coincides with the minimum-cost spanning tree problem (MST). In general, polynomial algorithms exist for $|V \setminus K|$ or $|K|$ bounded. The first, formulated by Hakimi [9] has complexity $O(2^{|V \setminus K|} \cdot |V|^2)$, the second is a dynamic programming algorithm of complexity $O(3^{|K|} \cdot |V| + 2^{|K|} \cdot |V|^2)$, due to Dreyfus and Wagner [5].

In the general case the branch and bound algorithm of Beasley [3] performs well. He formulates the SPG as a MST problem with additional constraints, that are relaxed in a Lagrangean fashion to obtain lower bounds; see also Duin and Volgenant [6].

Efficient reduction tests that reduce the values of $|V|$ and $|E|$ are valuable, since the running time of any SPG-algorithm or heuristic is related to these numbers. Additionally, heuristics may produce a better solution in the reduced graph. In Section II we reconsider known reduction tests and where possible improve them. The tests exploit $(d_{ij})_{i,j \in V}$ the matrix of shortest path 'lengths' with respect to $(c_{ij})_{i,j \in V}$, the cost matrix ($c_{ij} = c_{ji} = \infty$ for non existing edges). We give two examples from resp. Beasley [2] and Shore et al. [14].

—Edge $(i,j) \in E$ can be deleted if $d_{ij} < c_{ij}$.

—Vertex $i \in V \setminus K$ of degree two, say adjacent to p and q, can be eliminated if $d_{pq} < c_{ip} + c_{iq}$.

In Section III we consider the concept of 'bottleneck length', resulting in new effective reduction tests. Section IV is devoted to a general degree test and a transformation of edge costs. After discussing the implementation in Section V, the results in the last section illustrate the joint impact of the improved and new tests.

We will use the following terminology, definitions and notations:

—For any vertex $i \in V$, δ_i is the degree of i in G and σ_i is the minimum degree vertex i can have in a Steiner tree.

—As mentioned d_{ij} denotes the length of a shortest path between two vertices $i,j \in V$ with respect to the matrix (c_{ij}), whereas SP_{ij} stands for a set of edges—with maximum cardinality in case of choice—that form such a path.

—We will refer to a (Steiner) tree as a subset of E, of V, or as a subgraph of (V,E).

—A tree T' is called an *enlargement* of a tree T, if tree T' contains all the vertices of T.

—Edge $(i,j) \in E$ is a *Steiner edge* if it is part of some Steiner tree and if every Steiner tree T with $i,j \in T$ (can be enlarged so that it) contains edge (i,j) too. At any time in the process of solving an SPG, the symbol cT stands for the total cost of all Steiner edges detected so far.

—For typesetting reasons suffixes may be bracketed, e.g., k_{t+1} is written as $k(t + 1)$.

An edge (vertex) is eliminated, whenever a reduction test, say X, reveals that this edge (vertex) is not part of a Steiner tree. With phrases like: "Edge (i,j) is an X-edge" and "X is true for vertex i," we express that edge (i,j) resp. vertex i can be eliminated from the problem because of test X. On the other hand, if a test reveals that an edge,

say (i,j), is a Steiner edge, then (i,j) is incorporated in the solution and the following are performed:

- Raise cT by the addition of c_{ij} (and fuse vertex j into i as follows);
- For each $w \in V \setminus \{i\}$ adjacent to j, (re)define the edge (i,w) having cost $\min\{c_{iw}, c_{jw}\}$;
- Eliminate vertex j together with its edges.

In the reduced graph the new degree of i is: $\delta_i + \delta_j - 2 - |\{w \in V | (i,w) \in E \ \& \ (j,w) \in E\}|$. The d-matrix can be updated in $O(|V|^2)$ time, although for most reduction tests this is not necessary, see Section V.

2. REDUCTION TESTS AND IMPROVEMENTS

We consider existing reduction tests for the SPG and improve some of them. The first test already mentioned in the introduction, is almost trivial. It can only be effective in non-Euclidean graphs, when the edge costs violate the triangle inequality. In improved form, the test may also eliminate when the shortest path length equals the cost of the edge.

Least Cost Test (LC). Any edge $(i,j) \in E$ with $|SP_{ij}| > 1$ can be eliminated.
Clearly any vertex $i \in V \setminus K$ in a Steiner tree, must be adjacent to at least two other vertices, i.e., $\delta_i \geq \sigma_i \geq 2$, and a vertex $k \in K$ must have $\sigma_k \geq 1$. So both the following tests are valid.

Degree Tests (D1, D2). Any nonspecial vertex i with $\delta_i = 1$ or 2 can be eliminated. If $\delta_i = 2$, say adjacent to $p,q \in V$, then vertex i and the edges (p,i) and (i,q) are replaced by a single edge (p,q) with $c_{pq} = c_{pi} + c_{iq}$ if necessary, i.e., if $SP_{pq} = \{(p,i), (i,q)\}$.

Beasley [2] gives an important test to detect Steiner edges:

Nearest Vertex Test (NV). For any $k \in K$, let i be the nearest vertex, i.e., $c_{ki} = \min\{c_{kj} | j \in V\}$. Then (k,i) is a Steiner edge, if a vertex $k' \in K \setminus \{k\}$ exists such that

$$c_{ki} + d_{ik'} \leq \min\{c_{kj} | j \in V \setminus \{i\}\}.$$

The test holds trivially if $\delta_k = 1$ or if $i \in K$ $(k' = i)$. We prove a more general theorem.

Theorem 1. For an edge (i,j) to be a Steiner edge the following condition is sufficient: Two vertices $k,k' \in K$ exist such that $d_{kk'} = d_{ki} + c_{ij} + d_{jk'}$ and every path in $G \setminus \{(i,j)\}$ between k to k' contains an edge (s,t) of cost $c_{st} \geq d_{kk'}$.

Proof. Let T be a Steiner tree with $(i,j) \notin T$. In T there must be a path from k to k' without edge (i,j) on it. According to the assumption this path contains an edge (s,t) with $c_{st} \geq d_{kk'}$.

Enlarge T into the solution $T' = T \setminus \{(s,t)\} \cup S$, where S denotes a shortest path between k and k' with (i,j) on it. Then $(i,j) \in T'$ and T' is a Steiner tree, because:

$$c(T') \leq c(T) + d_{kk'} - c_{st} \leq c(T). \qquad \blacksquare$$

In Section III we develop the NSV test, that detects all edges satisfying the condition in $O(|V|^2)$ time.

If an upper bound structure U of cost $c(U)$ is available, e.g., by one of the heuristics mentioned in the Introduction, one can try to eliminate vertices and/or edges—not in U—that are "not reachable." That is, the cost of (a lower bound for) a solution tree containing the particular vertex (edge) exceeds the cost $c(U)$. Beasley [2] gives such a test, which we here improve:

Reachability Test (RT). For any $i \in V \setminus U$, let k' maximize d_{ik} for $k \in K$, $k0$ minimize d_{ik} for $k \in K$ and $k1$ minimize d_{ik} for $k \in K \setminus \{k0\}$. We can eliminate vertex i if

$$cT + d_{ik'} + d_{ik0} + d_{ik1} \geq c(U).$$

Under the RT-test condition vertex i has $\sigma_i = 0$ or 2. The latter case is only possible if $cT + d_{ik'} + d_{ik0} < c(U)$; in that case we add edges (p,q) for vertices p, q adjacent to i with $d_{pq} \geq c_{pi} + c_{iq}$. The original version of this test did not include the terms d_{ik0} and d_{ik1}.

Iwainsky et al. [10] use a cut based lower bound for the cost $c(S)$ of a solution S containing $i \in V \setminus U$. To sharpen their bound, suppose the tree S to be directed out of vertex i. Since each $k \in K$ has an incoming edge in S, the cost $c(S)$ must be at least cK defined as

$$cK = \sum_{k \in K} \underline{c}_k, \qquad \text{with } \underline{c}_k = \min\{c_{kt} | t \in V\}.$$

Furthermore there are two elementary paths from i to special vertices m and n (say) without other special vertices, but with one edge incident to m resp. n on them.

Cut Reachability Test (CRT). a) Vertex $i \in V \setminus U$ can be eliminated if

$$cT + cK + \min\{\underline{d}_{im} + \underline{d}_{in} | m,n \in K;\ m \neq n\} \geq c(U),$$

where d_{ik} denotes the value $d_{ik} - c_k$ for $k \in K$.

b) Edge $(i,j) \in E$ with $i \in V \setminus U$ can be eliminated if

$$cT + cK + \min\{\underline{d}_{im} + c_{ij} + \underline{d}_{jn} | m,n \in K;\ m \neq n\} \geq c(U).$$

Both versions are still $O(|K|)$ tests. It may happen that CRT fails on vertex $i \in V \setminus U$, while it is successful on edge (i,j), because for $m,n \in K$:

$$\underline{d}_{im} + \underline{d}_{in} \leq \underline{d}_{im} + c_{ij} + \underline{d}_{jn}.$$

3. A BOTTLENECK APPROACH AND RELATED TESTS

The Steiner problem in graphs is strongly related to the minimum spanning tree problem. The NV test exploits the fact that for every vertex a minimum-cost edge emanating from that vertex is in the MST. Unlike the LC test it uses some information about the position of the special vertices in the graph. The following new test (see Fig. 1) does also use such information.

Vertices Nearer to K Test (VNK). Edge (i,j) can be eliminated if a vertex $k \in K$ exists with $\max\{d_{ki}, d_{kj}\} < c_{ij}$.

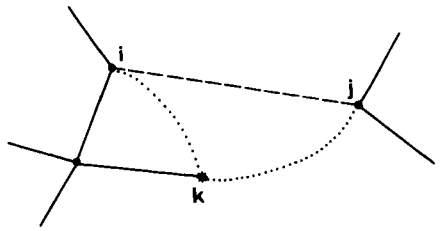

FIG. 1. Any solution with edge (i,j) is improved by replacing (i,j) with either SP_{ki} or SP_{kj}.

A LC-edge as well as a VNK-edge cannot be part of a MST on G. Both tests work locally in a δ-neighborhood of (i,j) with $\delta = c_{ij}$. In contrast to the LC test, VNK can eliminate edges in Euclidean graphs. If an edge is a VNK-edge for (V,K,E,c) then it can still be in the Steiner tree of (V,K',E,c), for a subset $K' \subseteq V$ other than K. This does not hold for an LC-edge, since it cannot be in the MST on any subgraph of G.

We now introduce some notions that lead to a characterization of Steiner edges and to a general edge elimination test. This test will use global information and has both LC and VNK as special cases.

—In a weighted graph $G = (V,E,c)$, the *bottleneck length* $bl(P_{ij})$ for a path P_{ij}, connecting vertices $i,j \in V$, is the maximum edge weight on the path, i.e.,

$$bl(P_{ij}) = \max\{c_{vw} | (v,w) \in P_{ij}\}.$$

—The *bottleneck distance* b_{ij} is the minimum bottleneck length over all paths P_{ij}, i.e.,

$$b_{ij} = \min\{bl(P_{ij}) | P_{ij} \text{ is a path connecting } i \text{ and } j\}.$$

—A set of vertices is called a *Steiner set* if a Steiner tree exists that contains this set.

—The distance graph $D(G)$ is the complete undirected graph with vertex set V and weight d_{ij} on edge (i,j) for each $i,j \in V$.

—A *D-path* is a path connecting vertices in $D(G)$. If the set of vertices on such a path is a Steiner set then it is called a *Steiner path*. A D-path is called a *special path* if all intermediary vertices (if any) are special.

—The *Steiner distance* σ_{ij} is the minimum bottleneck length in $D(G)$ over all Steiner paths, i.e.,

$$\sigma_{ij} = \min\{\max\{d_{vw} | (v,w) \in Q_{ij}\} | Q_{ij} \text{ is a Steiner path between } i \text{ and } j\}.$$

As a consequence $\sigma_{ij} = \infty$ *if* $\{i,j\}$ is not a Steiner set.

—The *special distance* s_{ij} is the minimum bottleneck length over all special paths in $D(G)$, i.e.,

$$s_{ij} = \min\{\max\{d_{vw} | (v,w) \in Q_{ij}\} | Q_{ij} \text{ is a special path between } i \text{ and } j\}.$$

Resuming all values defined for $i,j \in V$, we have: $b_{ij} \leq s_{ij} \leq d_{ij} \leq c_{ij}$, and if set $\{i,j\}$ is a Steiner set: $b_{ij} \leq \sigma_{ij} \leq s_{ij} \leq d_{ij} \leq c_{ij}$.

It is easy to prove that the bottleneck distance b_{ij} equals the bottleneck length of the elementary path connecting i and j in a MST of (V,E,c). So a necessary and sufficient condition for some edge (i,j) to be part of an MST is: $b_{ij} = c_{ij}$. The next theorem translates this to the SPG.

Theorem 2. Edge $(i,j) \in E$ is a Steiner edge if and only if $\sigma_{ij} = c_{ij}$.

Proof. Let (i,j) be a Steiner edge. Then $\sigma_{ij} \leqslant c_{ij}$, since $\{i,j\}$ must be a Steiner set. Suppose $\sigma_{ij} < c_{ij}$, i.e., a Steiner path $Q_{ij} = \{(i,i_1), (i_1,i_2), \ldots, (i_n,j)\}$ exists with

$$\max\{d_{vw} | (v,w) \in Q_{ij}\} < c_{ij}.$$

There must be a Steiner tree T that contains both the Steiner set $\{i = i_0,i_1, \ldots, i_n, j = i_{n+1}\}$ and the Steiner edge (i,j). The graph $T \setminus \{(i,j)\}$ consists of two components T_i and T_j containing i respectively j. If m $(\leqslant n)$ is the maximum index with $i_0,i_1,i_2, \ldots, i_m \in T_i$, then a shortest path $SP_{i(m)i(m+1)}$ connects T_i to T_j at a cost $d_{i(m)i(m+1)} < c_{ij}$. Thus

$$T' = T \setminus \{(i,j)\} \cup SP_{i(m)i(m+1)}$$

is a better solution than T, contradicting the optimality of T.

To prove the other direction, let $\sigma_{ij} = c_{ij}$ for an edge $(i,j) \in E$. Since $\{i,j\}$ must then be a Steiner set, an optimal solution T exists with $i,j \in T$. Suppose edge (i,j) itself is not (yet) in T. There must be a path Q_{ij}, connecting i and j in T. Since T is optimal, Q_{ij} is a Steiner path and has an edge (v,w) in it of cost $c_{vw} = d_{vw} \geqslant c_{ij}$. Thus $T' = \{(i,j)\} \cup T \setminus \{(v,w)\}$ is a Steiner tree that is an enlargement of T—see definitions in the introduction—and (i,j) is a Steiner edge. ∎

Unfortunately, as we do not know which D-paths are Steiner paths, the relation $\sigma_{ij} = c_{ij}$ cannot be checked easily. However, the bottleneck distance b_{ij} is a lower bound for σ_{ij}. For instance, for edges (k,k') with $k,k' \in K$, the Steiner distance $\sigma_{kk'}$ is at most $c_{kk'}$. We have equality if the NV test is true on $k \in K$ with nearest vertex $i = k' \in K$, since then $b_{kk'} \geqslant \min\{c_{kj} | j \in V\} = c_{kk'}$. Also the Demand Node Aggregation test of Balakrishnan and Patel [1]—stating that an edge (k,k') with $k,k' \in K$ is a Steiner edge, if it is in a MST of (V,E)—is equivalent with $b_{kk'} = c_{kk'}$. Both these tests are incorporated in our Steiner edge test, which inspects the weaker condition of Theorem 1. To make an efficient check possible the condition is reformulated in the following lemma. The proof is straightforward but rather technical so we omit it.

Lemma 1. Edge $(i,j) \in E$ satisfies the condition of Theorem 1 if and only if

$$\underline{b}_{k1k2} \geqslant d_{k1i} + c_{ij} + d_{jk2}, \tag{1}$$

where $\underline{b}$ denotes the bottleneck length in $\underline{G} = G \setminus \{(i,j)\}$ and $k1, k2 \in K$ are chosen such that

$$d_{k1i} = \min\{d_{ki} | k \in K, d_{ki} < d_{kj}\} \quad \text{and} \quad d_{k2j} = \min\{d_{kj} | k \in K, d_{kj} < d_{ki}\} \tag{2}$$

We will deduce an $O(|V|^2)$ time complexity for detecting all edges satisfying the above condition. In fact, it can only be true for MST-edges. To show this, suppose that (i,j)

cannot be in a MST on G, and yet is a Steiner edge according to Theorem 1, i.e., for some vertices $k, k' \in K$

$$\underline{b}_{kk'} \geq d_{ki} + c_{ij} + d_{jk'} = d_{kk'}.$$

Then the first assumption implies $\underline{b}_{kk'} = b_{kk'}$ and because either $b_{kk'} < d_{kk'}$ or $SP_{kk'} = \{(k,k')\}$, we deduce from the second assumption that $SP_{kk'} = \{(k,k')\} = \{(i,j)\}$. But then $b_{kk'} = c_{kk'}$, so $(k,k') = (i,j)$ can be part of a MST on G, a contradiction.

It takes time $O(|V|^2)$ to find a minimum spanning tree T on G. Assuming the shortest path distances are already known, the values $d_{k1i} + c_{ij} + d_{k2j}$, for the edges $(i,j) \in T$—with $k1$, $k2$ satisfying (2)—can be calculated in $O(|V| \cdot |K|)$ operations. In order to verify the inequality (1) for an edge (i,j) of a MST, we only need to compare the length c_{vw} of the shortest chord (v,w) that replaces branch (i,j) in a MST on $\underline{G}$, with the memorized value $d_{k1i} + c_{ij} + d_{k2j}$ (for a proof, see the NSV test below). It is well known that all branch chord exchanges only consume $O(|V|^2)$ time. We conclude that detecting all Steiner edges satisfying Theorem 1, requires $O(|V|^2)$ time.

Nearest Special Vertices Test (NSV). For edge (i,j) in a MST on G, let chord (v,w) replace branch (i,j) in a MST on $\underline{G} = G \setminus \{(i,j)\}$; furthermore let $k1$, $k2$ satisfy (2). Then (i,j) is a Steiner edge if $c_{vw} \geq d_{k1i} + c_{ij} + d_{k2j}$.

Proof. We will show that the above condition is equivalent to condition (1) of Lemma 1.

Suppose (1) is true for $(i,j) \in T$, a MST on G and let $\underline{T}$ be a MST on $\underline{G}$. First we note that $k1$ and $k2$ are in different components of $T \setminus \{(i,j)\}$. Otherwise

$$d_{k1k2} \geq b_{k1k2} = \underline{b}_{k1k2} \geq d_{k1i} + c_{ij} + d_{k2j} \geq d_{k1k2},$$

so $k1 = i$ and $k2 = j$ contradicting the assumption that $k1$ and $k2$ are in the same component. We conclude that $(v,w) \in \underline{T}_{k1k2}$, the path connecting $k1$ to $k2$ in $\underline{T}$. If some edge $(r,s) \in \underline{T}_{k1k2}$ has cost $c_{rs} > c_{vw}$ then $(T \setminus \{(r,s)\}) \cup \{(v,w)\}$ would be a spanning tree on G of lower cost than T; thus

$$c_{vw} = bl(\underline{T}_{k1k2}) = \underline{b}_{k1k2} \geq d_{k1i} + c_{ij} + d_{k2j}.$$

On the other hand, if $c_{vw} \geq d_{k1i} + c_{ij} + d_{k2j}$, then again the assumption that $k1$ and $k2$ are in the same component of $T \setminus \{(i,j)\}$—say for instance the component containing i—leads to a contradiction: since also $d_{k2j} < d_{k2i}$, this would imply

$$d_{k2j} = \underline{d}_{k2j} \geq \underline{b}_{k2j} = bl(\underline{T}_{k2j}) \geq c_{vw} > d_{k2j}.$$

So because $(v,w) \in \underline{T}_{k1k2}$ we have: $\underline{b}_{k1k2} = bl(\underline{T}_{k1k2}) \geq c_{vw} \geq d_{k1i} + c_{ij} + d_{k2j}$. ■

With similar but more elaborate proofs one can show that Lemma 1 as well as the NSV test remain valid if in their formulation all shortest path distances are replaced by corresponding special distances. So the test can be performed just as well in case only the special distances are available; it then has the same effect and complexity.

As a corollary of Theorem 1, we can eliminate some edge (i,j) if $\sigma_{ij} \neq c_{ij}$. This is equivalent to $\sigma_{ij} < c_{ij}$ or $\sigma_{ij} = \infty$. A reachability test proves the latter, when demonstrating that $\{i,j\}$ is not a Steiner set. If such tests fail, the other option to

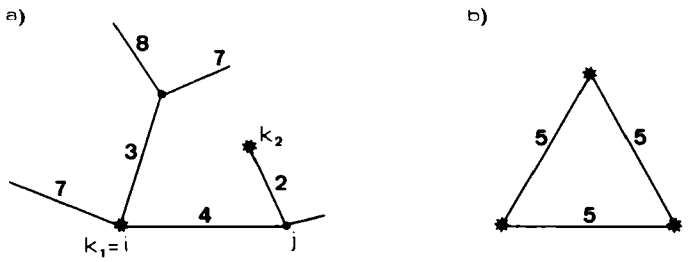

FIG. 2. (a) The NSV test finds optimal edge (i,j), whereas NV and Demand Node Aggregation do not. (b) All three Steiner edge tests pick out two Steiner edges.

eliminate edge (i,j) is to prove: $\sigma_{ij} < \infty$ implies $\sigma_{ij} < c_{ij}$. This is done by the following test. Instead of using a lower bound on σ_{ij}, the above is derived by means of the upper bound s_{ij} on σ_{ij} $(<\infty)$.

Smaller Special Distance Test (SD). Any edge $(i,j) \in E$ can be eliminated if $s_{ij} < c_{ij}$.

This test reduces to the LC resp. VNK test, when only D-paths with zero, resp. one intermediary special vertex are considered. Moreover the SD test generalizes the so called "*R-R*, *R-S* and *S-S* edge deletion" methods of Balakrishnan and Patel [1], see Duin and Volgenant [7].

Though edge (i,j) can also be eliminated if a nontrivial special path $Q_{ij}(\neq \{(i,j)\})$ exists with $bl(Q_{ij}) = c_{ij}$, there is a complication. In a repeated application without updating the matrix (s_{ij}) of special distances in the graph, allowance of equality may cause illegal eliminations, e.g., in the graph of Figure 2b all edges are eliminated. Nevertheless, for integer edge costs, distributed with small variance and/or over a small range, it is attractive to consider the equality sign in SD. Then one can change equalities into inequalities by defining a suitable perturbation on cost function c.

4. A GENERAL DEGREE TEST AND AN EDGE COST TRANSFORMATION

The degree tests D1 and D2 eliminate vertices with degree up to 2. We now deal with vertices of larger degree, exploiting the special distances.

Let $i \in V \setminus K$, with $\sigma_i = 3$ be adjacent to vertices n_1, n_2, n_3, by edges of cost c_1, c_2, and c_3 respectively (see Fig. 3). Whenever we have shown that $\sigma_i < 3$, we can eliminate i, because then one can simply add new edges (n_1,n_2), (n_2,n_3), and (n_1,n_3) of costs respectively $c_1 + c_2$, $c_2 + c_3$ and $c_1 + c_3$, to account for the case of $\sigma_i = 2$ ($\sigma_i > 1$ as $i \in V \setminus K$). Of course SD edges among these three are not added. Indeed $\sigma_i < 3$, if $d_{n1n2} + d_{n1n3} \leq c_1 + c_2 + c_3$. Since, any solution S with $\sigma_i = 3$ can be transformed into another solution S' with $\sigma_i \leq 2$ and $c(S') \leq c(S)$, by replacing the edges incident to i for SP_{n1n2} and SP_{n1n3}.

The above condition, implying $\sigma_i = 0$ or 2, can be loosened by using special distances instead of shortest path lengths; also it can be made suitable for vertices $i \in V \setminus K$ of arbitrary degree. For this purpose let G_i denote the complete undirected

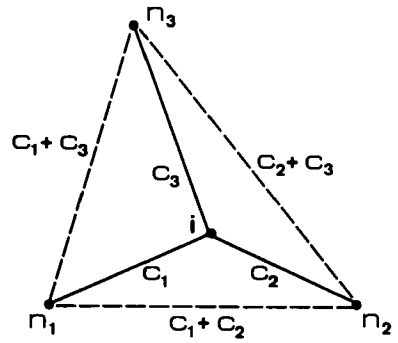

FIG. 3. The BD3 test eliminates $i \in V \setminus K$, while 0, 1, or 2 of the dotted edges are added.

graph on vertex set $V_i = \{j \in V | j \text{ is adjacent to } i\}$ weighted with special distances s_{pq} for $p,q \in V_i$. Suppose $V_i = \{n_1,n_2,n_3\}$, then the left hand side in the condition $s_{n1n2} + s_{n1n3} \leq c_{in1} + c_{in2} + c_{in3}$ can be replaced by Z_i^*, the value of a minimum spanning tree on G_i. We obtain the following general degree test.

Bottleneck Degree *m* Test (BD*m*). For $i \in V \setminus K$ with $\delta_i = m$, test BD$m(i)$ is true, i.e., vertex i can be eliminated if

> for every subset V' of V_i with $|V'| \geq 3$ the weight Z' of an MST on the induced subgraph of V' in G_i satisfies $Z' \leq \Sigma_{j \in V'} c_{ij}$.

When i is eliminated, edges (p,q) of cost $c_{pi} + c_{iq}$ are added for $p,q \in V_i$ with $s_{pq} \geq c_{pi} + c_{iq}$.

Proof. Being trivially true for $m = 1$ and 2, BDm comprises D1 and D2. For $m > 2$ it is sufficient to show that $\sigma_i = 0$ or 2, since the insertion of edges (p,q) for $p,q \in V_i$ takes care of the latter case.

Now let S^* be a Steiner tree containing vertex i in such a way that the total cost of the edges incident to i is at minimum and suppose $\delta_i(S^*) = n > 2$. Define $V' = V_i \cap S^*$ and let T denote an MST on the induced subgraph of V' in G_i with cost $Z' \leq \Sigma_{j \in V'} c_{ij}$. Consider S, initially defined as $S^* \setminus \{i\}$; its cost is then $c(S^*) - \Sigma_{j \in V'} c_{ij}$ and it consists of n components $S_1, \ldots, S_n$. For ease of notation we say that $V' = \{1,2, \ldots, n\}$ with node $j \in S_j$. Using "edges" of T we can restore the connectivity between these components for a cost of at most Z', by repeating the following operation:

Take a shortest "edge" from T; say (v,w) is shortest, $s_{vw} = \min\{s_{pq} | (p,q) \in T\}$. Let a special path of bottlenecklength s_{vw} be given by $SQ_{vw} = \{k_0 = v, k_1, \ldots, k_r = w\}$. Add to S the shortest paths $SP_{k(t)k(t+1)}$ for the indices t with k_t and k_{t+1} in different components of S. If a shortest path links more than two components it is partitioned into subpaths (each having only its endnodes in different components of S) and these subpaths are successively added. Since v and w are in different components of S at least one such (sub)path is inserted. After the addition of a (sub)path—linking two components, say S_p and S_q, for a cost of at most s_{vw}—we transform the tree T as follows. A branch on the elementary path connecting p to q in T is eliminated and in T vertices p,q are fused.

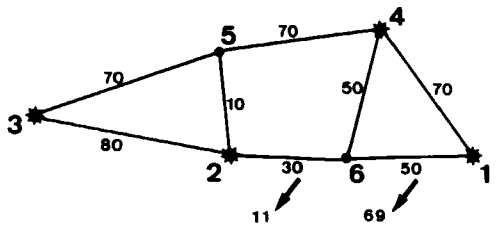

FIG. 4. In (V,K,E,c') reduction is possible, while it is not in (V,K,E,c).

In this way each addition of a (sub)path simultaneously reduces the number of branches in the tree T and the number of components in S by one, at a cost not larger than the special distance of the deleted branch. The operation can be repeated until S is a spanning tree, which is equivalent to T being empty. The total cost then spent on the $n - 1$ additions of (sub)paths is at most $\Sigma_{(p,q)\in T} s_{pq} = Z'$, so in case of $Z' < \Sigma_{j\in V'} c_{ij}$ we have contradicted the optimality of S^*.

If $Z' = \Sigma_{j\in V'} c_{ij}$ and $c(S) = c(S^*)$, then (because of the positive edge costs and the assumption that $\Sigma_{j\in V'} c_{ij}$ is at minimum for Steiner tree S^*) the set of edges added to $S^* \setminus \{i\}$ by the $n - 1$ subpaths is exactly $\{(i,j) | j \in V_i\}$. So $\delta_i(S) = n$ and every edge (p,i) for $p \in V'$ was involved in a (sub)path. This path must also contain another edge (i,q) incident with vertex i. But then at most $[n/2]$ different (sub)paths can be added for a cost Z'. So S must still consist of at least $n - [n/2]$ components; a contradiction, since $n > 2$. ∎

For BD3 the condition only needs to be checked once for $V' = V_i$. When true—for a vertex i with $V_i = \{n_1,n_2,n_3\}$—at most two new edges are added; because of the positive edge costs $s_{n1n2} + s_{n1n3} \leq c_{n1i} + c_{n2i} + c_{n3i}$ implies $s_{n1n2} < c_{n1i} + c_{n2i}$ or $s_{n1n3} < c_{n1i} + c_{n3i}$. Though polynomial for m fixed, the effort for BDm grows exponentially with m; therefore we only implemented it for $m \leq 4$.

It is their cooperation and repetition that makes the described tests successful. A transformation of the edge costs can sometimes act as an animator of further reductions. Consider the graph in Figure 4 with $K = \{1,2,3,4\}$. None of the tests help in (further) reducing this graph. Suppose we shift an amount of cost $\alpha = 19$ from edge (2,6) to edge (6,1), i.e., $c'_{26} := 30 - \alpha = 11$ and $c'_{61} := 50 + \alpha = 69$. We shall see that the new problem is equivalent to the original one. Now (further) reducing is possible: the NSV test detects Steiner edge (2,6); after identifying vertex 6 with 2, the BDm test eliminates vertex 5 without adding any edges; finally the problem is solved by three applications of the NSV test.

Proposition. Let (t,i) and (i,j) be two edges incident with $i \in V \setminus K$ that are both Steiner edges for the SPG on $(V, K \cup \{i\}, E, c)$. Further assume that both edges are indispensable Steiner edges for the SPG on $(V, K \cup \{i\}, E, c')$, where c' equals c, except:

$$c'_{ti} = c_{ti} + \alpha \quad \text{and} \quad c'_{ij} = c_{ij} - \alpha, \quad \text{with } -c_{ti} \leq \alpha \leq c_{ij}.$$

Then any Steiner tree T on (V,K,E,c') is also optimal for the original problem.

Proof. We first show that $c(T) = c'(T)$.

Case 1. $i \in T$: Then T is also optimal on $(V, K \cup \{i\}, E, c')$ and both edges (t,i) and (i,j), being indispensable Steiner edges, are in T, e.g., $c'(T) = c(T)$.

Case 2. $i \notin T$: Then edges (t,i) and (i,j) are not in T and again $c'(T) = c(T)$.

Now suppose T is not optimal on (V,K,E,c), but tree S with $c(S) < c(T)$ is. In case $i \in S$, if necessary we enlarge S to contain both (t,i) and (i,j), then analogously: $c(S) = c'(S)$. This contradicts the optimality of T on (V,K,E,c'), because $c'(S) = c(S) < c(T) = c'(T)$. ∎

If the NSV test fails on adjacent edges (k,i) and (i,j), it can be attractive to transform edge costs.

Change Edge-Costs (CEC). Consider two edges (k,i) and (i,j) in a MST on G with $k \in K \setminus \{j\}$ and $i \in V \setminus K$. If the NSV test is true on (i,j) for the SPG on $(V, K \cup \{i\}, E, c)$, then one can change their edge costs:

$$c'_{ki} := c_{ki} + \alpha \quad \text{and} \quad c'_{ij} := c_{ij} - \alpha,$$

where $\alpha \leqslant c_{ij}$ and $\alpha < \underline{b}_{ki} - c_{ki}$ with $\underline{b}$ the bottleneck length in $G \setminus \{(k,i)\}$.

Immediate reduction after CEC is possible if $c'_{ij} = 0$ (fusion of i and j) or if $c'_{ki} \geqslant c_{kq}$ and $q \in K$, where q denotes the (originally) second nearest vertex to k, since edge (k,q) is then a NSV-edge.

5. IMPLEMENTATION OF REDUCTION TESTS

In order to investigate the effectiveness of the described tests, we give a reduction algorithm that applies these tests repeatedly. For the sake of efficiency, the application is not always exhaustive; it is interrupted if the computational effort per reduced edge/vertex becomes too large.

In the implementation of the given tests, there are possibilities for an efficient approach. The LC test can be performed repeatedly with the original (d_{ij}) matrix, as this matrix is invariant under LC elimination. This is not trivial for the SD test in general, as entries of the d-matrix can increase because of SD elimination. Fortunately, as a corollary of the following lemma, it is correct to reduce a graph by applying in any order NSV, BDm, and SD tests with respect to the special matrix (s_{ij}) of the original graph (V,E).

Lemma 2. If graph $(V,E,[s_{ij}])$ is reduced to graph $(V',E',[s'_{ij}])$ by application of tests NSV, BDm, and/or SD, then for vertices $i,j \in V \cap V'$, the special distance has not increased, i.e., $s'_{ij} \leqslant s_{ij}$.

Proof. Let $s_{ij} = bl(Q_{ij})$ for the special path $Q_{ij} = \{(i = k_0,k_1), \ldots, (k_n,k_{n+1} = j)\}$. It suffices to prove that $s'_{ij} \leqslant s_{ij}$ if just one of the mentioned tests (not eliminating vertex i or j) is applied.

NSV: The fusion of two vertices can only decrease shortest path lengths, so clearly $s'_{ij} \leqslant s_{ij}$.

BDm: A degree test can only eliminate a nonspecial vertex p. Also the shortest path lengths remain unchanged, if we act as though for all $v,w \in V_p$ edges (v,w) are added,

eliminating edges (v,w) with $s_{vw} \geq c_{pv} + c_{pw}$ by means of SD. Thus the special path Q_{ij} in (V,E) still exists in (V',E') with unchanged bottleneck distance, i.e., $s'_{ij} = s_{ij}$.

SD. Suppose an edge $(s,t) \in \mathrm{SP}_{k(r)k(r+1)}$ is eliminated by SD for some $r \leq n$. Then a special path Q_{st} exists with (say) intermediary vertices $v_1, v_2, \ldots, v_m$ $(m \geq 1)$ such that

$$\max\{d_{vw} | (v,w) \in Q_{st}\} < c_{st} \leq bl(Q_{ij}).$$

Consider the special path in $D(V',E')$ along vertices $k_1, \ldots, k_r, v_1, \ldots, v_m, k_{r+1}, \ldots, k_{n+1}$. Because

$$d_{k(r)v(1)} \leq d_{k(r)s} + d_{sv(1)} \leq d_{k(r)s} + c_{st} \leq d_{k(r)k(r+1)} \leq bl(Q_{ij})$$

and analogously $d_{v(m)k(r+1)} \leq bl(Q_{ij})$, its bottleneck length equals $bl(Q_{ij})$, so $s'_{ij} = s_{ij}$. ■

The shortest path matrix (d_{ij}) can be calculated with a $|V|$-fold version of Dijkstra's [8] algorithm that is of complexity $O(|V|^3)$. When matrix (d_{ij}) is available, an additional effort of order $O(|V|^2 \cdot |K|)$ is required for the matrix (s_{ij}). The time for calculating these matrices is large, compared to the time needed to perform reduction tests on every edge (vertex) of the graph. For the tests using special distances, we will exploit the fact that any upper bound on these values is also usable, since they check a $\leq -$ relation. Thus applying SD, BDm, and NSV repeatedly with, e.g., c_{ij} as upper bound on s_{ij}, one reduces the graph in order to calculate the matrix (s_{ij}) more cheaply. Examining also special paths made of two edges, a sharper upper bound is:

$$\min[c_{ij}, \min\{c_{it} \oplus c_{tj} | t \in V_i \cap V_j\}], \tag{3}$$

with

$$\begin{aligned} c_{ti} \oplus c_{tj} &= \max\{c_{ti}, c_{tj}\} && \text{if } t \in K, \\ &= c_{ti} + c_{tj} && \text{otherwise.} \end{aligned}$$

Initially, the algorithm repeats the following operations as long as they are successful:

(re)calculation of upper bounds on s_{ij}, first $s_{ij} = c_{ij}$ and further $s_{ij} = (3)$; SD, BDm $(m \leq 3)$ and NSV.

For dense graphs, c.q. $|E| > V.\ln|V|$, the above is not rerun as only the upper bounds c_{ij} are efficient.

Next the matrix (s_{ij}) is calculated and the following sequence of tests is cycled:

SD	(including the equality case with a suitable perturbation);
BDm	(for $m \leq 4$ and repeated exhaustively)
NSV	(this time with extension: after each fusion the matrix (s_{ij}) is updated, and BDm is applied whenever degrees on nodes are reduced)
CEC	(after this test the s-matrix has to be recalculated).

The arrangement of the tests is logical. Preprocessing NSV by BDm cannot disturb the detection of Steiner edges but only enhance it, as the associated elimination and

TABLE I. The reduction process on three problems (—not done; | unchanged).

Problem type	Sparse random				Sparse Euclidean				Full dense Euclidean			
	|V|	|K|	|E|		|V|	|K|	|E|		|V|	|K|	|E|	
Problem size	75	13	150	%cpu	200	50	400	%cpu	100	25	4950	%cpu
Initialization				14.3				13.8				14.6
Calc. $d_{ij} \setminus s_{ij}$		(upper bounds)		2.8		(upper bounds)		1.1		(exact values)		35.3
SD	|	|	146	.3	|	|	397	.1	|	|	308	4.8
Degree tests	56	|	127	1.4	151	|	346	.8	77	|	253	.0
NSV	49	10	120	1.7	122	29	309	2.2	68	17	243	3.9
CEC	—	—	—	—	—	—	—	—	64	13	236	.0
Calc. $d_{ij} \setminus s_{ij}$		(upper bounds)		.5		(upper bounds)		.8		(exact values)		4.3
SD	|	|	114	.2	|	|	295	.1	|	|	|	.4
Degree tests	47	|	112	.7	105	|	267	.6	57	|	214	.0
NSV	46	9	111	.9	103	28	264	.8	|	|	|	.0
CEC	—	—	—	—	—	—	—	—	54	11	208	.1
Calc. $d_{ij} \setminus s_{ij}$		(upper bounds)		.7		(upper bounds)		.4		(exact values)		3.0
SD	|	|	|	.2	|	|	256	.1	|	|	|	.3
Degree tests	|	|	|	.5	100	|	252	.2	|	|	|	.0
NSV	|	|	|	.9	|	|	|	.1	53	10	207	.1
CEC	45	|	110	.7	94	23	246	.8	|	|	|	.0
Calc. $d_{ij} \setminus s_{ij}$		(exact values)		52.1		(exact value)		58.1				
SD	|	|	81	6.9	|	|	161	6.5				
Degree tests	16	|	22	3.3	42	|	69	.7				
NSV	7	4	11	5.2	30	16	52	4.4				
CEC	6	|	10	.9	25	13	44	.4				
Calc. $d_{ij} \setminus s_{ij}$		(exact values)		.7		(exact values)		2.5				
SD	|	|	7	.5	|	|	40	.9				
Degree tests	4	|	3	.0	18	|	27	.1				
NSV		problem solved		.7	14	9	23	.4				
CEC					11	6	20	.3				
Calc. $d_{ij} \setminus s_{ij}$						(exact values)		.4				
SD					|	|	16	.1				
Degree tests					8	|	10	.1				
NSV					6	4	8	.2				
CEC						problem solved		.1				
Total time		0.4 sec				2.7 sec				14.0 sec		

lengthening of edges may increase bottleneck distances. Performing the SD test before the BDm degree test is also sensible, since it lowers the degrees making an efficient BDm possible (for $m \leq 4$). After an exhaustive application of the NSV test it is likely (because of its extension) that also the tests SD and BDm are exhausted, so the situation calls for an edge cost transformation.

As soon as the above cycle of tests is without sufficient reduction, it is first extended with the reachability tests RT and CRT. Since they are not likely to succeed in an unreduced graph it is prudent to postpone these tests. Moreover each time before their application the matrix d_{ij} has to be (re)calculated, an $O(|V|^3)$ effort that one does not want to waste. The upper bound $c(U)$ was calculated according to Takahashi and Matsuyama [15].

When also the extended cycle becomes unsuccessful a lower bound is computed via the ascent method of Beasley [3], with a maximum number of 300 Lagrangean dual subproblems. The ascent procedure is not called upon as long as other tests are successful, in order to solve the $O(|V|^2)$ subproblems in a reduced graph. Because each subproblem is an MST problem, branch chord exchanges (for short BCE) constitute an additional reduction test, that is important only in case of a small duality gap. If the BCE tests are successful and the gap is non-zero, the entire procedure is once repeated; for the sake of efficiency the program terminates before entering a second ascent.

To give an impression of the share in time and reduction of the different tests and procedures, as well as their cooperation, we recorded in Table I the course of the reduction program on three problems of different type (a description of these types is given in the next section). Only the full dense Euclidean problem invoked the (C)RT tests and the Lagrangean dual ascent with BCE tests; both were without an effect but absorbed 0.3 resp. 29.2% of total cpu time.

6. COMPUTATIONAL RESULTS

We give computational results for the given tests and the transformation CEC on many problems. In Table II we compare our results with Beasley [3] for the problems until size 100, that are generated by him and made available to us. These problems have integer edge costs, uniformly drawn on the interval [1, 10]. We similarly created the other problems of size 200. The memory space on our computer did not permit us to jump from size 100 to 500, like Beasley did. For his results the solution times are obtained with Fortran on a Cray 1 supercomputer and for our results with Pascal on a Cyber 175. As far as we know the former is at least two times faster than the latter and Fortran more than twice as fast as Pascal. We multiplied the Cray 1 times with a conservative factor of 4 to obtain the fairly comparable times in Table II.

To show the effect of the reduction tests we give the size of the graph (V',E') after reduction, before and where necessary (between brackets) after a dual Lagrangean lower bound ascent combined with BCE. For the same values of $|V|$, $|E|$, and $|K|$, we generated Euclidean problems with integer edge costs taken as the rounded up distances between randomly drawn vertices in a square of size 100. Table III gives also results on full density problems, both Euclidean and random; the latter have integer edge

TABLE II. Comparison of results for randomly drawn problems until size 200 (#MST: number of MST problems solved in Lagrangean ascent; * problem solved; — not available; ( ) result after ascent with branch chord exchanges.)

Problem size			Beasley's results				New results			
|V|	|E|	|K|	|V'|	|E'|	#MST	cpu	|V'|	|E'|	#MST	cpu
50	63	9	(*)	(*)	73	.52	*	*	0	.06
		13	(*)	(*)	22	.40	*	*	0	.07
		25	(*)	(*)	4	.40	*	*	0	.08
	100	9	(23)	(35)	162	1.92	*	*	0	.20
		13	(*)	(*)	69	1.12	*	*	0	.10
		25	(*)	(*)	38	.60	*	*	0	.15
75	94	13	(*)	(*)	11	.72	*	*	0	.10
		19	(*)	(*)	13	.72	*	*	0	.10
		38	*	*	0	.68	*	*	0	.11
	150	13	(*)	(*)	98	2.20	*	*	0	.42
		19	(*)	(*)	293	3.84	*	*	0	.26
		25	(*)	(*)	189	2.20	*	*	0	.17
100	125	17	(*)	(*)	94	2.40	*	*	0	.25
		25	(*)	(*)	266	3.88	15 (*)	22 (*)	24	.39
		50	(*)	(*)	20	1.56	*	*	0	.18
	200	17	(79)	(166)	172	6.28	38 (*)	77 (*)	38	2.38
		25	(*)	(*)	581	10.12	*	*	0	.47
		50	(6)	(7)	169	2.52	*	*	0	.26
200	400	34	—	—	—	—	59 (*)	110 (*)	43	8.21
		50	—	—	—	—	*	*	0	2.54
		100	—	—	—	—	*	*	0	.75

costs uniformly drawn in the interval [1, 100]. The computation times refer to total cpu time in seconds inclusive of input and output.

The contribution of the newly developed tests and techniques is shown by giving separately the performance of a version of the program from which the tests BDm, NSV, SD, and CEC as well as the upper bound technique on the special distances were cut out. In cases where a (proven) optimal value was unknown, the cost of the best known feasible solution was used calculating the duality gap. We comment on the results:

—The newly developed tests and techniques, reduce problems in general further; the computation times compare favorably, up to a factor ten for sparse problems.

—The program performs better for larger values of $|K| \setminus |V|$. This can be explained: The value s_{ij} then is a better upper bound on σ_{ij} ($<\infty$); also the ascent produces a better lower bound.

—Problems are often completely solved by reduction tests only, especially if:

$$|E| \leq 2 \cdot |V| \quad \text{or} \quad |K| \geq |V|/2$$

or if edge costs are uniformly random edge costs.

—Though improved, the reachability tests RT and CRT were unsuccessful. On the

TABLE III. Results on Euclidean problems (upper part) and random problems (lower part). #MST: number of MST problems solved in Lagrangean ascent; * problem solved; in case of ( ) the ascent with BCE had no effect. The gap (after the Lagrangean ascent) is given as: 100% − (best lower bound)/(optimal value) × 100%.

Problem size			Without new tests					With new tests				
$\|V\|$	$\|E\|$	$\|K\|$	$\|V'\|$	$\|E'\|$	#MST	gap	cpu	$\|V'\|$	$\|E'\|$	#MST	gap	cpu
50	63	9	*	*	0	0	.3	*	*	0	0	.1
		13	13 (*)	22 (*)	16	0	.4	*	*	0	0	.1
		25	9 (*)	15 (*)	74	.3	.5	*	*	0	0	.1
	100	9	37 (37)	83 (83)	137	2.8	2.6	14 (12)	28 (22)	213	.5	.9
		13	43 (*)	92 (*)	48	0	1.7	*	*	0	0	.3
		25	12 (*)	27 (*)	11	0	.5	*	*	0	0	.1
75	94	13	17 (*)	31 (*)	7	0	.9	*	*	0	0	.1
		19	*	*	0	0	.8	*	*	0	0	.1
		38	14 (*)	25 (*)	10	0	.9	*	*	0	0	.1
	150	13	56 (15)	125 (28)	173	1.3	6.7	*	*	0	0	.4
		19	51 (18)	118 (35)	263	.1	9.9	10 (*)	16 (*)	11	0	.4
		38	35 (19)	89 (35)	95	.4	2.7	*	*	0	0	.4
100	125	17	25 (10)	40 (15)	74	.80	2.3	*	*	0	0	.3
		25	15 (*)	26 (*)	15	0	1.7	*	*	0	0	.2
		50	*	*	0	0	1.6	*	*	0	0	.2
	200	17	75 (*)	173 (*)	67	0	5.8	27 (*)	48 (*)	76	0	1.9
		25	63 (41)	157 (98)	300	1.8	15.3	11 (*)	18 (*)	16	0	.8
		50	54 (*)	129 (*)	46	0	4.4	*	*	0	0	.3

Problem size			Without new tests					With new tests				
$\|V\|$	$\|E\|$	$\|K\|$	$\|V'\|$	$\|E'\|$	#MST	gap	cpu	$\|V'\|$	$\|E'\|$	#MST	gap	cpu
200	400	34	156 (84)	352 (166)	226	.4	79.3	*	*	0	0	7.3
		50	129 (*)	310 (*)	74	0	26.6	*	*	0	0	2.7
		100	98 (*)	270 (*)	68	0	19.6	8 (*)	12 (*)	8	0	1.9
50	1225	9	47 ( )	1019 ( )	67	64.9	5.9	37 ( )	158 ( )	161	25.3	4.1
		13	49 ( )	1179 ( )	207	15.1	14.2	34 ( )	99 ( )	147	7.8	3.6
		25	40 ( )	395 ( )	179	6.9	5.9	*	*	0	0	1.6
75	2775	13	73 ( )	2303 ( )	300	31.4	42.8	63 ( )	245 ( )	158	19.0	9.9
		19	70 ( )	1744 ( )	295	22.8	33.7	44 ( )	160 ( )	250	7.1	9.2
		38	55 ( )	940 ( )	223	7.5	16.5	*	*	0	0	4.5
100	4950	17	98 ( )	4114 ( )	67	74.4	32.1	83 ( )	407 ( )	138	34.5	17.9
		25	95 ( )	3710 ( )	67	63.5	30.2	53 ( )	207 ( )	185	13.7	14.0
		50	82 ( )	1477 ( )	195	10.5	29.7	21 ( )	58 ( )	169	2.2	10.7
50	1225	9	34 (*)	87 (*)	10	0	1.9	*	*	0	0	1.1
		13	44 (*)	136 (*)	246	0	7.9	*	*	0	0	1.2
		25	11 (*)	21 (*)	6	0	1.1	*	*	0	0	1.3
75	2775	13	68 (*)	222 (*)	25	0	6.5	*	*	0	0	3.2
		19	65 (35)	202 (71)	169	0	11.5	*	*	0	0	3.3
		38	39 (*)	123 (*)	15	0	3.7	*	*	0	0	4.1
100	4950	17	95 (*)	359 (*)	102	0	18.6	44 (*)	89 (*)	28	0	9.4
		25	82 (*)	131 (*)	30	0	12.1	*	*	0	0	7.5
		50	51 (*)	146 (*)	5	0	8.1	*	*	0	0	8.8

considered problems practically the same reduction could have been achieved without them. Only in cases with $|K'| < 6$, did we record some successes for the CRT test.

—The great difference in cpu time between sparse problems and solved full dense problems of the same size $|V|$, is mainly due to the primary use of upper bounds on (s_{ij}) in sparse problems (thus calculating the exact values in a reduced graph), and to some extent also to the use of a linked list instead of a matrix to store the set of edges.

We described a number of improved and new reduction tests for the SPG. Among them three strong tests excel, namely the tests NSV, SD, and BDm. By the use of a min max measure, the special distance s_{ij}, they surpass the local character of formerly applied tests.

While remaining efficient, the optimal edge detector "Nearest Special Vertices" generalizes both the "Nearest Vertex" test and the "Demand Node Aggregation" test. The edge eliminator "Special Distance" incorporates the deletion methods given in Balakrishnan and Patel [1], as well as the "Least cost" test. The BDm can eliminate many redundant nodes; moreover it may increase bottleneck distances, thus enhancing the performance of NSV.

We conclude that the special distances deserve a prominent role when solving the Steiner problem in graphs. They induce efficient tests that reduce the problem graph substantially, at a small computational price. A branch and bound algorithm including the tests NSV, SD, and BDm is most promising. We think it is even worth considering to develop such an algorithm without a (Lagrangean) lower bound, only using reduction tests.

References

[1] A. Balakrishnan and N. R. Patel, Problem reduction methods and a tree generation algorithm for the Steiner network problem. *Networks* **17** (1987) 65–85.

[2] J. E. Beasley, An algorithm for the Steiner problem in graphs. *Networks* **14** (1984) 147–159.

[3] J. E. Beasley, An SST-based algorithm for the Steiner problem in graphs. Report of the Department of Management Science, Imperial College, London, England (1985).

[4] E. J. Cockayne, On the efficiency of the algorithm for Steiner minimal trees. *SIAM Appl. Math.* **18** (1970) 150–159.

[5] S. E. Dreyfus and R. A. Wagner, The Steiner problem in graphs. *Networks* **1** (1972) 195–207.

[6] C. W. Duin and A. Volgenant, Some generalizations of the Steiner problem in graphs. *Networks* **17** (1987) 353–364.

[7] C. W. Duin and A. Volgenant, An edge elimination test for the Steiner problem in graphs. Operation Research Letters **8** (1989) 79–83.

[8] E. W. Dijkstra, A note on two problems in connexion with graphs. *Numerische Math.* **1** (1959) 269–271.

[9] S. L. Hakimi, Steiner's problem in graphs and its applications. *Networks* **1** (1971) 113–133.

[10] A. Iwainsky, E. Canuto, O. Taraszow, and A. Villa, Network decomposition for the optimization of connection structures. *Networks* **16** (1986) 205–235.

[11] N. Maculan, The Steiner problem in graphs. *Ann. Discrete Math.* **31** (1987) 185–212.

[12] Z. A. Melzak, On the problem of Steiner. *Canad. Math. Bull.* **4** (1961) 143–148.

[13] V. J. Rayward-Smith and A. Clare, On finding Steiner vertices. *Networks* **16** (1986) 283–294.

[14] M. L. Shore, L. R. Foulds, and P. B. Gibbons, An algorithm for the Steiner problem in graphs. *Networks* **12** (1982) 323–333.

[15] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs. *Math. Japonica* **24** (1980) 573–577.
[16] P. Winter, The Steiner problem in networks: A survey. *Networks* **17** (1987) 129–167.

Received October, 1986
Accepted August, 1988