Chapter 4

Heuristics

As already mentioned in Chapter 1, the decision version of the Steiner tree problem is NP-complete. Consequently, no polynomial time algorithm for the Steiner tree problem is likely to exist. In view of this inherent intractability of the problem, it is of practical importance to develop heuristics that quickly find low-length trees spanning the set of terminals N. In this chapter available heuristics for the Steiner tree problem are reviewed. Section 4.1 covers path heuristics. They gradually add appropriately chosen paths between a tree constructed so far and terminals not yet in the tree. Tree heuristics are described in Section 4.2. They start with a tree spanning all terminals. Various strategies are then applied to obtain a shorter tree. Vertex heuristics are discussed in Section 4.3. Their common characteristic is that they select a subset of "good" non-terminals. Once they are selected, a minimum spanning tree of the subnetwork induced by terminals and selected non-terminals yields a suboptimal solution. Heuristics that do not fall into any of the above three classes are discussed in the subsequent sections: contraction heuristic in Section 4.4, dual ascent heuristic in Section 4.5 and set covering heuristic in Section 4.6. Finally, computational experience and comparison of heuristics is mentioned in Section 4.7.

4.1 Path Heuristics

Several heuristics for the Steiner tree problem can be characterized as path heuristics. Starting from an arbitrarily chosen terminal (or some other subnetwork of G), the tree is gradually grown until it spans all terminals. The expansion is typically based on the addition of (shortest) paths between vertices already in the tree and terminals not yet in the tree.

4.1.1 Shortest Paths Heuristic

Takahashi and Matsuyama [32] suggested a heuristic for the Steiner tree problem related to Prim's minimum spanning tree algorithm [27]: when a partial tree containing a subset N_k of terminals has been built up, an appropriately chosen

terminal $z_{k+1} \notin N_k$ is connected to it by a shortest path. More specifically, the *shortest paths heuristic* (SPH) is as follows.

- **Step 1:** Begin with a subtree T_{SPH} of G consisting of a single, arbitrarily chosen, terminal z_1 (Fig. 4.1a). $k = 1$.
- **Step 2:** If $k = n$, then **Stop**.
- **Step 3:** Determine a terminal $z_{k+1} \notin T_{SPH}$ closest to T_{SPH} (ties are broken arbitrarily). Add to T_{SPH} a shortest path joining it with z_{k+1} (Fig. 4.1b and Fig. 4.1c). $k = k + 1$. Go to **Step 2**.

As noted by Rayward-Smith and Clare [29], T_{SPH} can be further improved by two additional steps.

- **Step 4:** Determine a minimum spanning tree for the subnetwork of G induced by the vertices in T_{SPH} (Fig. 4.1d).
- **Step 5:** Delete from this minimum spanning tree non-terminals of degree 1 (one at a time). The resulting tree is the (suboptimal) solution. **Stop**.

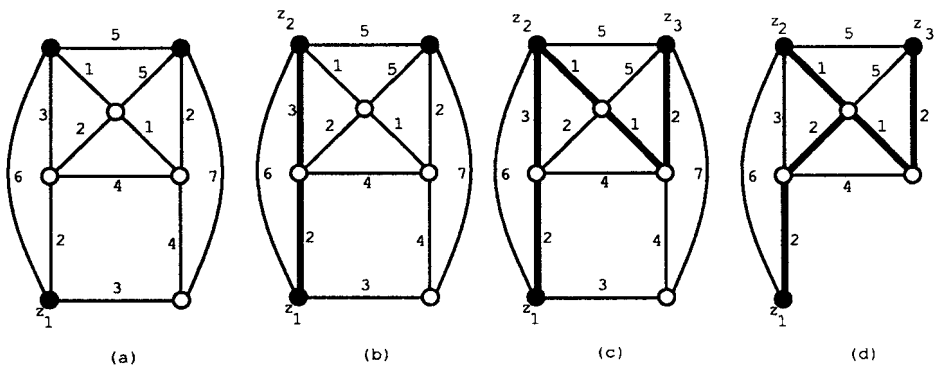

Figure 4.1: Shortest paths heuristic

The shortest paths heuristic can be implemented by a straightforward modification of Prim's algorithm for minimum spanning trees [27], given shortest paths from every terminal to all other vertices. Furthermore, the determination of these shortest paths is the bottleneck of the heuristic. Consequently, the shortest paths heuristic requires $O(nv^2)$ time, since shortest paths from each terminal to all other vertices can be determined by for example Dijkstra's algorithm [8] in $O(v^2)$ time. Alternatively, use of Fibonacci heaps to compute shortest paths [12] reduces the worst-case time complexity of the shortest paths heuristic for sparse networks to $O(n(e + v \log v))$.

A very important issue associated with heuristics for the Steiner tree problem is how bad can suboptimal solutions be in comparison with optimal solutions. Let $T_G(N)$ denote a Steiner minimal tree for N in G. Let L denote a walk around $T_G(N)$ traversing each edge exactly twice. One can regard L as composed of n *terminal paths*, each connecting a terminal to a "next" terminal (Fig. 4.2). The

longest terminal path P^* of L must have length at least $2|T_G(N)|/n$. When P^* is deleted from L, a walk L' with $|L'| \leq |T_G(N)|(2-2/n)$ is obtained.

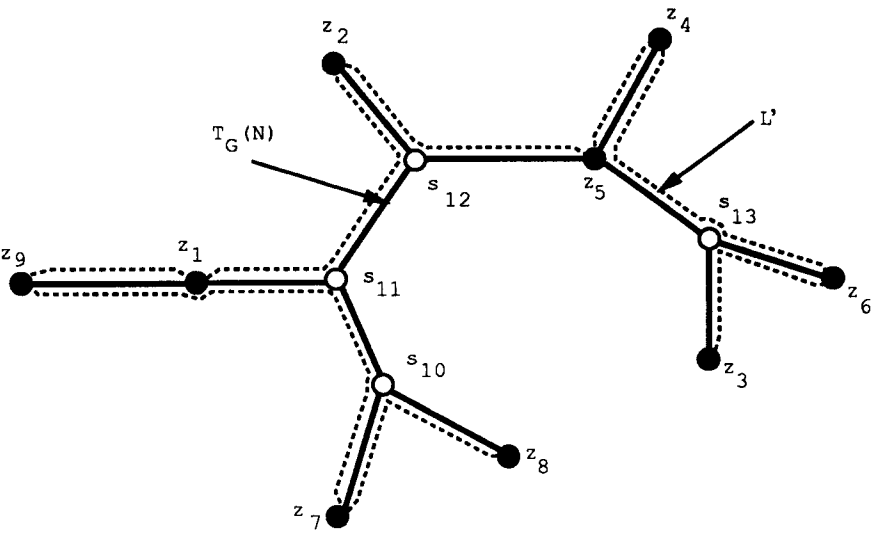

Figure 4.2: Walk L' around $T_G(N)$

Theorem 4.1 ([32]) *$|T_{SPH}|/|T_G(N)| \leq 2-2/n$ for any network G and any set of terminals N. Furthermore, this bound is tight, i.e., for any ε, $\varepsilon > 0$, there is an instance of the Steiner tree problem such that $|T_{SPH}|/|T_G(N)| > 2-\varepsilon$.*

Proof. In order to prove the first part of the theorem, one needs to show that $|T_{SPH}| \leq |L'|$. Let $z_1, z_2, ..., z_n$ be the ordering in which the terminals are selected by the shortest paths heuristic. Let P_k, $k = 2, 3, ..., n$, denote the shortest paths used to construct T_{SPH}. It will be shown that to each P_k corresponds a different, not shorter terminal path in L'. For each terminal z_k, $2 \leq k \leq n$:

- traverse L' backward (counterclockwise) beginning at z_k until reaching the first terminal path, denoted by B_k, from a terminal z_{k_b}, $k_b < k$ to a terminal z_i, $i \geq k$. Let $k_b = 0$ if B_k does not exist.
- traverse L' forward (clockwise) beginning at z_k until reaching the first terminal path, denoted by F_k, from a terminal z_j, $j \geq k$, to a terminal z_{k_f}, $k_f < k$. Let $k_f = 0$ if F_k does not exist.

Note that at least one of the paths B_k and F_k exists. Let L_k denote B_k if $k_b > k_f$ and F_k otherwise. Referring to Fig. 4.2 where $P^* = \{z_3, s_{13}, z_5, s_{12}, s_{11}, s_{10}, z_8\}$:

$$B_2 = \{z_1, s_{11}, s_{12}, z_2\}, F_2 = \emptyset, L_2 = B_2$$

$$B_3 = \{z_2, s_{12}, z_5\}, F_3 = \emptyset, L_3 = B_3$$

$$B_4 = \{z_2, s_{12}, z_5\}, F_4 = \{z_6, s_{13}, z_3\}, L_4 = F_4$$

$$B_5 = \{z_2, s_{12}, z_5\}, F_5 = \{z_5, z_4\}, L_5 = F_5$$
$$B_6 = \{z_4, z_5, s_{13}, z_6\}, F_6 = \{z_6, s_{13}, z_3\}, L_6 = B_6$$
$$B_7 = \emptyset, F_7 = \{z_9, z_1\}, L_7 = F_7$$
$$B_8 = \emptyset, F_8 = \{z_8, s_{10}, z_7\}, L_8 = F_8$$
$$B_9 = \{z_7, s_{10}, s_{11}, z_1, z_9\}, F_9 = \{z_9, z_1\}, L_9 = B_9$$

Let k and k' be two distinct integers satisfying $2 \leq k < k' \leq n$. Suppose that z_k precedes $z_{k'}$ on L'. The opposite case is proved in similar manner. Assume first that all terminals between z_k and z'_k on L' have indices higher than k. Then $B_{k'}$ begins somewhere between z_k and $z_{k'}$. Furthermore, B_k either does not exist or begins at a terminal preceding z_k. Hence $B_k \neq B_{k'}$. There are three possibilities for $F_{k'}$:

- $F_{k'}$ does not exist. Then F_k does not exist. Hence, $L_k = B_k$ and $L_{k'} = B_{k'}$. As already remarked, $B_k \neq B_{k'}$. Thus, $L_k \neq L_{k'}$.
- $F_{k'}$ ends in a terminal with index $k'_f > k$. F_k either does not exist, or ends in a terminal with index k_f satisfying $k'_f > k > k_f$. So both $B_k \neq B_{k'}$ and $F_k \neq F_{k'}$, implying that $L_k \neq L_{k'}$.
- $F_{k'}$ ends in a terminal with index $k'_f < k$. In this case, $F_k = F_{k'}$. However, $k'_b > k$ implying that $L_{k'} = B_{k'}$. Since $B_{k'} \neq B_k$ and $B_{k'} \neq F_k$, it follows again that $L_k \neq L_{k'}$.

Assume next that there is at least one terminal between z_k and $z_{k'}$ with index less than k. Then it follows immediately that $B_k, F_k, B_{k'}, F_{k'}$ are all distinct. Consequently, $L_k \neq L_{k'}$.

So far it has been proved that the paths $L_2, L_3, ..., L_n$ are mutually distinct. Each path L_k, $k = 2, 3, ..., n$, goes from a terminal with index less than k to a terminal with index greater than or equal to k. Consequently, L_k is not shorter than the path to z_k selected by the shortest paths heuristic.

It remains to show that the bound is tight. Consider the network shown in Fig. 4.3. Its n terminals form a path with edges of length 2. Its only non-terminal is adjacent to all terminals. All edges incident with the non-terminal have length $1 + \delta$, $\delta > 0$. T_{SPH} is the path through all terminals and $|T_{SPH}| = 2n - 2$. For sufficiently small δ, $T_G(N)$ consists of the edges incident with the non-terminal and $|T_G(N)| = n + n\delta$. As δ goes to 0, the ratio goes to $2 - 2/n$. For sufficiently large n, the ratio will be greater than $2 - \varepsilon$. □

4.1.2 Repetitive Shortest Paths Heuristics

The shortest paths heuristic is sensitive to the choice of the initial vertex from which the solution is constructed. Thus, it can be worthwhile to apply the shortest paths heuristic more than once, each time beginning with a different initial vertex (or subnetwork). Several such repetitive strategies have been investigated by Winter and Smith [39].

- SPH-N: determine T_{SPH} n times, each time beginning with a different terminal.
- SPH-V: determine T_{SPH} v times, each time beginning with a different vertex.

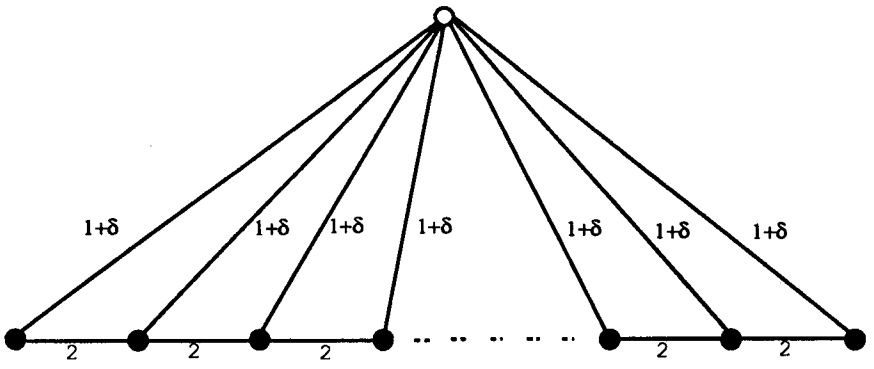

Figure 4.3: Tightness of the error ratio bound of T_{SPH}

- SPH-zN: determine T_{SPH} $n-1$ times, each time beginning with a shortest path from a fixed terminal z_1 to a terminal z_i, $i = 2, 3, ..., n$.
- SPH-NN: determine T_{SPH} $n(n-1)/2$ times, each time beginning with a shortest path between a different pair of terminals.

The repetitive applications of the shortest paths heuristic will yield better solutions. In particular, the SPH-V and SPH-NN variants seem to perform well. The improvement in terms of quality of obtained solutions is of course paid for by longer computational times. However, the repetitive schemes become much more attractive if the reductions described in Chapter 2 are applied first.

4.1.3 Shortest Paths with Origin Heuristic

Takahashi and Matsuyama [32] also studied the *shortest paths with origin heuristic* (SPOH). T_{SPOH} consists of the shortest paths from an arbitrarily chosen terminal z_i to all other terminals. The worst-case time complexity of this heuristic is $O(e + v \log v)$ using Fibonacci heaps. The worst-case error ratio $|T_{SPOH}|/|T_G(N)|$ is tightly bounded by $n-1$. Hence, the shortest paths with origin heuristic can be considered as inferior to the shortest paths heuristic also in the worst-case sense. It performs poorly on average.

4.1.4 Kruskal-Based Heuristic

The shortest paths heuristic (Subsection 4.1.1) is closely related to Prim's algorithm for minimmum spanning trees [27]. Wang [35] suggested a heuristic which is closely related to Kruskal's algorithm for minimum spanning trees. The *Kruskal-based heuristic* (KBH) is as follows.

- **Step 1:** Begin with a forest T_{KBH} consisting of all isolated terminals.
- **Step 2:** If T_{KBH} is connected, then **Stop**.
- **Step 3:** Find two trees in T_{KBH} closest to each other (ties are broken arbitrarily). Add to T_{KBH} a shortest path connecting those two trees. Go to **Step 2**.

The worst-case time complexity of this heuristic is $O(nv^2)$. Widmayer [37] showed that $|T_{KBH}|/|T_G(N)| \leq 2 - 2/n$. Plesník [25] showed that this bound is tight in the same sense as the bound for the error ratio of T_{SPH}.

4.1.5 Y-Heuristic

A variant of the shortest paths heuristic was suggested by Chen [5]. It will be referred to as the *Y-heuristic* (YH). Rather than begin with an arbitrary terminal, Steiner minimal trees for all triplets of terminals are determined. Such Steiner minimal trees consist of at most one non-terminal of degree 3 from which three paths to the terminals leave. The topology of such Steiner minimal trees can be represented by the letter Y. Hence the name of the heuristic. The shortest Steiner minimal tree taken over all triples of terminals is used as a starting tree (instead of an arbitrary terminal chosen by the shortest paths heuristic).

The expansion phase of the Y-heuristic is also slightly different than in the shortest paths heuristic. Suppose that a tree T_k spanning k terminals, $3 \leq k < n$, has been constructed. A terminal $z_{k+1} \notin T_k$ closest to T_k is determined. Let $v_{k+1} \in T_k$ denote the other end-vertex of the corresponding shortest path. Let q denote the degree of v_{k+1} in T_k. T_k contains q edge-disjoint elementary paths, all beginning at v_{k+1} and with intermediate non-terminals (if any) of degree 2. Each such elementary path is removed in turn. This splits T_k into two components (temporarily contracted to two terminals). They are reconnected together with z_{k+1} using the Steiner minimal tree construction for triplets. The shortest among the q trees generated is selected as the tree T_{k+1} spanning $k+1$ terminals to be used in the next iteration.

Chen [5] showed that a Steiner minimal tree for a triplet of terminals can be determined in $O(e \log v)$ time. Furthermore, a shortest Steiner minimal tree for all $O(n^3)$ triplets of terminals can be found in $O(ne \log v)$ time. Each iteration of the Y-heuristic involves the determination of $O(n)$ Steiner minimal trees for triplets of vertices. Before a shortest among these $O(n)$ Steiner minimal trees is determined, up to $O(v)$ edges will be temporarily contracted or deleted from G in $O(e)$ time. Hence, the k-th iteration requires $O(ne \log v + e) + O(e + v \log v)$ time where the second term is the time needed to identify z_{k+1}. This reduces to $O(ne \log v)$. There are $n-3$ iterations. Hence, the Y-heuristic requires in total $O(n^2 e \log v)$ time. Not surprisingly, on average, solutions are better than those obtained by the shortest paths heuristic. Widmayer [38] showed that $|T_{YH}|/|T_G(N)| \leq 2 - 2/n$. Plesník [25] showed that this bound is tight in the same sense as the bound for the error ratio of T_{SPH}.

4.2 Tree Heuristics

Another class of heuristics is based on the idea of constructing a tree spanning all terminals. Usually a variant of the minimum spanning tree algorithm is used to obtain this initial tree. Once given, various strategies to improve it can be applied.

4.2.1 Minimum Spanning Tree Heuristic

In the *minimum spanning tree heuristic* (MSTH) suggested by Takahashi and Matsuyama [32], the solution T_{MSTH} is obtained by deleting from the minimum spanning tree for G non-terminals of degree 1 (one at a time). The worst-case time complexity of the minimum spanning tree heuristic is $O(e+v\log v)$. The worst-case error ratio $|T_{MSTH}|/|T_G(N)|$ is tightly bounded by $v-n+1$. Hence, the minimum spanning tree heuristic can be considered as inferior to the shortest paths heuristic in the worst-case sense. It also performs poorly on average.

4.2.2 Greedy Tree Heuristic

Minoux [23] suggested a *greedy tree heuristic* (GTH). It constructs various minimum spanning trees of subnetworks of G induced by appropriate supersets of N. In order to ensure existence of such minimum spanning trees, G should be complete. If this is not the case, G can be transformed into a complete network G^* by the addition of infinite length edges as explained in Subsection 1.4.2. Alternatively, the problem can be solved in the distance network $D = D_G(V)$ rather than in G. However, this will involve the determination of all shortest paths. The heuristic is as follows.

- **Step 1:** Let $k = 0$ and $N_k = N$. Let T_k denote a minimum spanning tree of the subnetwork of G^* induced by N_k.
- **Step 2:** If $N_k = V$, then **Stop**. Otherwise, determine minimum spanning trees $T_k^{v_i}$ of subnetworks of G^* induced by $N_k \cup v_i$, $v_i \notin N_k$. Let $T_k^{v_j}$ denote the shortest of these minimum spanning trees.
- **Step 3:** If $|T_k| \leq |T_k^{v_j}|$, then let $T_{GTH} = T_k$ and **Stop**. Otherwise, $N_{k+1} = N_k \cup v_j$, $T_{k+1} = T_k^{v_j}$, $k = k+1$, and go to **Step 2**.

The number of iterations is $O(v-n)$. The number of minimum spanning trees determined during each iteration is also $O(v-n)$. These minimum spanning trees do not need to be determined from scratch. In fact, given a minimum spanning tree of a subnetwork of G^* induced by a vertex set N_k, $0 \leq k < v-n$, any minimum spanning tree of a subnetwork of G^* induced by a vertex set $N_k \cup v_i$, $v_i \notin N_k$, can be determined in $O(|N_k|)$ time [23]. It follows that the complexity of the greedy tree heuristic is $O((v-n)^2 v + n^2)$. If the greedy tree heuristic is solved in D, additional $O(v^3)$ time is required to determine all shortest paths. Note that the heuristic can completely fail to find a solution if T_0 contains infinite length edges. Even when solved in D, the heuristic can halt with T_0 as T_{GTH}. Hence, its worst-case error ratio is $v-n+1$.

Minoux [23] noticed that the efficiency of the greedy tree heuristic can be considerably improved if each pair of non-terminals is non-adjacent in the network G. Let $N_0 \subset N_1 \subset \ldots \subset N_k$, $0 \leq k < v-n$, be a sequence of subsets generated by the heuristic. Let $v_i \notin N_k$. Let $T_0^{v_i}, T_1^{v_i}, \ldots, T_k^{v_i}$ denote minimum spanning trees induced by $N_j \cup v_i$, $j = 0, 1, \ldots, k$. It can be shown [23] that

$$|T_0^{v_i}| - |T_0| \leq |T_1^{v_i}| - |T_1| \leq \ldots \leq |T_k^{v_i}| - |T_k|$$

The modified heuristic is then as follows.

- **Step 1:** Let $k = 0$ and $N_k = N$. Let T_0 denote a minimum spanning tree of the subnetwork of G^* induced by N_k.
- **Step 2:** If $N_k = V$, then **Stop**. Otherwise determine minimum spanning trees T^{v_i} of subnetworks of G induced by $N_k \cup v_i$, $v_i \notin N_k$. Place these trees on a queue Q in non-decreasing order of their length.
- **Step 3:** Let T^{v_j} denote the first minimum spanning tree on Q. Remove T^{v_j} from Q.
- **Step 4:** If T^{v_j} does not span $N_k \cup v_j$, then redefine T^{v_j} to be the minimum spanning tree of the subnetwork of G induced by $N_k \cup v_j$, add it to Q (preserving the non-decreasing order), and go to **Step 2**.
- **Step 5:** If $|T_k| \leq |T^{v_j}|$, then let $T_{GTH} = T_k$ and **Stop**. Otherwise, $N_{k+1} = N_k \cup v_j$, $T_{k+1} = T^{v_j}$, $k = k + 1$, and go to **Step 2**.

Computational experience reported by Minoux [23] indicates that the number of minimum spanning trees generated by the modified version is reduced approximately 4 times. However, the worst-case time complexity of the modified heuristic remains unchanged.

4.2.3 Distance Network Heuristic

The *distance network heuristic* (DNH) was suggested independently by Choukhmane [6], Kou, Markowsky and Berman [19], Plesník [24], and Iwainsky, Canuto, Taraszow and Villa [16]. It is as follows.

- **Step 1:** Construct the distance network $D_G(N)$ for N in G (Fig. 4.4a and Fig. 4.4b).
- **Step 2:** Determine a minimum spanning tree of $D_G(N)$ (indicated by heavy line segments in Fig. 4.4b).
- **Step 3:** Replace each edge in the minimum spanning tree by a corresponding shortest path in G (Fig. 4.4c). Let T_D denote this network. Note that shortest paths can be selected in such a way that T_D is a tree.
- **Step 4:** Determine a minimum spanning tree T_{DNH} of the subnetwork of G induced by the vertices of T_D (Fig. 4.4d).
- **Step 5:** Delete from T_{DNH} non-terminals of degree 1 (one at a time). **Stop.**

The overall worst-case time complexity of the distance network heuristic is $O(nv^2)$. It is Step 1 that dominates all the remaining steps. It involves solving n shortest path problems. The bound can be reduced for sparse networks to $O(n(e + v \log v))$ if Fibonacci heaps are used [12].

Theorem 4.2 *$|T_{DNH}|/|T_G(N)| \leq 2 - 2/n$ for any network G and any set N of terminals. Furthermore, this bound is tight, i.e., for any ε, $\varepsilon > 0$, there is an instance of the Steiner tree problem such that $|T_{DNH}|/|T_G(N)| > 2 - \varepsilon$.*

Proof. Let L be the walk around $T_G(N)$ as defined in Subsection 4.1.1. One may regard L as composed of at most n leaf-connecting paths. If the longest of these paths is deleted from L, a walk L'' such that $|L''| \leq |T_G(N)|(2 - 2/n)$ is obtained. On the other hand, $|T_{DNH}| \leq |T_D| \leq |L''|$, completing the proof of the first part of the theorem.

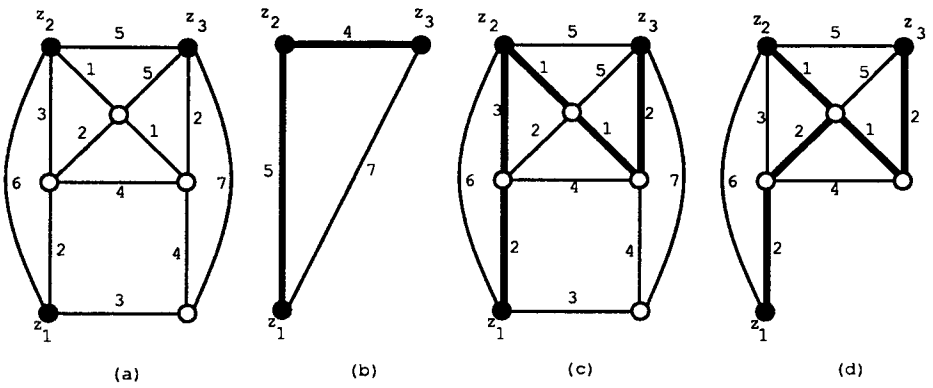

Figure 4.4: Distance network heuristic

The second part follows immediately using the same networks as when proving the tightness of the bound for the shortest paths heuristic in Subsection 4.1.1. □

Goemans and Bertsimas [15,14] proved a stronger result by showing that $|T_G(N)|$ in Theorem 4.2 can be replaced by $v(\mathcal{LP}_4)$; the optimal solution value of the linear relaxation of the mathematical programming formulation $\mathcal{P}_4$ of the Steiner tree problem (see Subsection 3.6.4).

The error bound for the distance network heuristic and for the shortest paths heuristic is the same. However, the latter often (but not always) produces better solutions. For instance, T_{SPH} indicated in Fig. 4.5a by heavy line segments has length 15 (if either z_1 or z_2 is chosen as the initial terminal) while T_{DNH} in Fig. 4.5b has length 16. On the other hand, T_{SPH} in Fig. 4.5c has length 45 while T_{DNH} in Fig. 4.5d has length 39.

It has been shown by Kucera, Marchetti-Spaccamela, Protasi and Talamo [21] that the distance network heuristic is nearly optimal for random graphs (i.e., networks where an edge of unit length is present with probability p). In that case the expected length of optimal solutions is approximately $n \log v / \log(vp)$. But even one of the most trivial heuristics, namely the shortest paths with origin heuristic (Subsection 4.1.3) is also nearly optimal for random graphs.

The inferiority of the solutions obtained by the distance network heuristic when compared with the shortest paths heuristic, together with the fact that their worst-case error ratios and worst-case time complexities are comparable, makes the distance network heuristic a poor choice among the heuristics for the Steiner tree problem. However, as will be seen in the remainder of this subsection, the worst-case time complexity of the distance network heuristic can be reduced substantially.

Wu, Widmayer and Wong [41], and independently Wang [35] suggested a method of determining T_D directly from G without explicit determination of the distance network $D_G(N)$. The construction of $D_G(N)$ is avoided by simultaneously growing shortest path trees from each terminal. During each iteration, the shortest, not yet

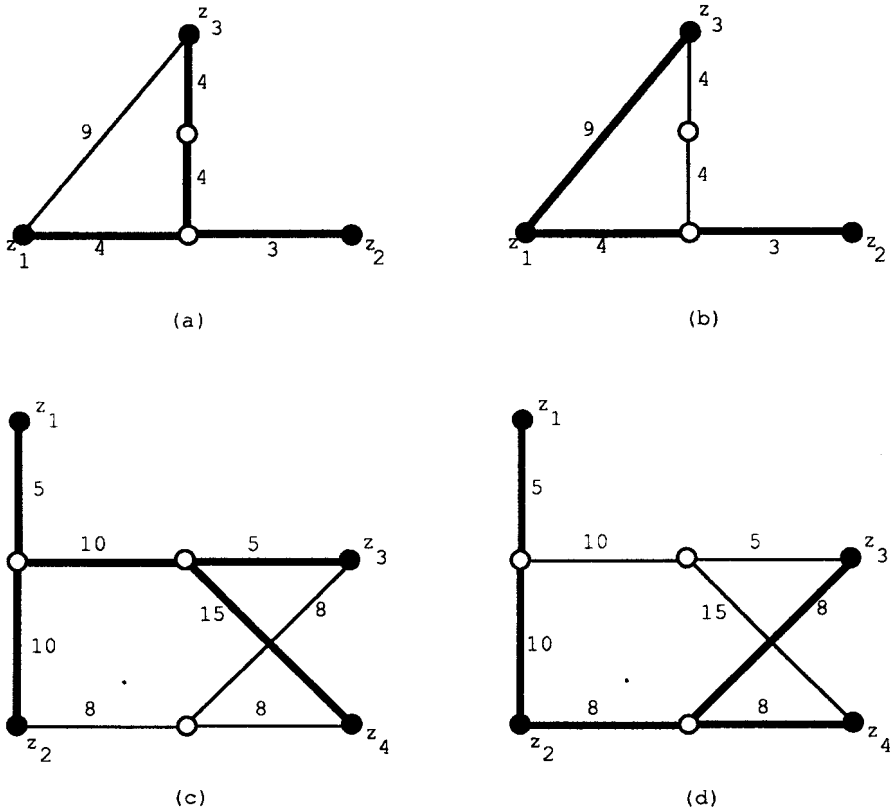

Figure 4.5: T_{SPH} outperforming T_{DNH} and vice versa

included, edge incident to one of the shortest path trees is added. When two trees meet, a path between their roots is formed (corresponding to a path in T_D). Two joined shortest path trees are restructured to a single tree with one root, and the expansion continues (until only one tree is left). The shortest path trees computation requires $O(e \log v)$ time. This is an improvement over the bound $O(nv^2)$ of the distance network heuristic, unless the network is dense and has few terminals. Unfortunately, the penalty is the need of rather complex data structures.

Further worst-case time complexity reduction when determining T_D, based on even more complex data structures [12,13] was given by Widmayer [37]. The worst-case time complexity of the distance network heuristic is then $O(e + (v + \min\{e, n^2\}) \log v)$. Recently, Kou [17] provided yet another algorithm for T_D. In the worst-case, his algorithm runs in $O(e + n \log n + (v - n) \log(v - n))$ time. If G is sparse, the time complexity reduces to $O((v - n) \log(v - n) + q \log \beta(q, n))$ where $q = \min\{e, (n-1)^2/2\}$ and $\beta(x, y) = \min\{i | \log^i y \leq x/y\}$.

Mehlhorn [22] suggested yet another modification of the distance network heuristic. Furthermore, this modification requires very simple data structures and has

superior worst-case time complexity. T_D is determined not from the distance network $D_G(N)$, but from another, easier to find, network G'. More specifically, G' is obtained as follows. Let $N(z_i), z_i \in N$, denote non-terminals of G closer to z_i than to any other terminal (Fig. 4.6a). Consider the network $G' = (N, E', c')$ where

$$E' = \{(z_i, z_j) \mid \exists (v_k, v_l) \in E : v_k \in N(z_i), v_l \in N(z_j)\}$$

and the length $c'(z_i, z_j)$ of the edge (z_i, z_j) in G' is

$$\min\{d(z_i, v_k) + c(v_k, v_l) + d(v_l, z_j) \mid (v_k, v_l) \in E, v_k \in N(z_i), v_l \in N(z_j)\}$$

The network G' obtained from G (Fig. 4.6a) is shown in Fig. 4.6b.

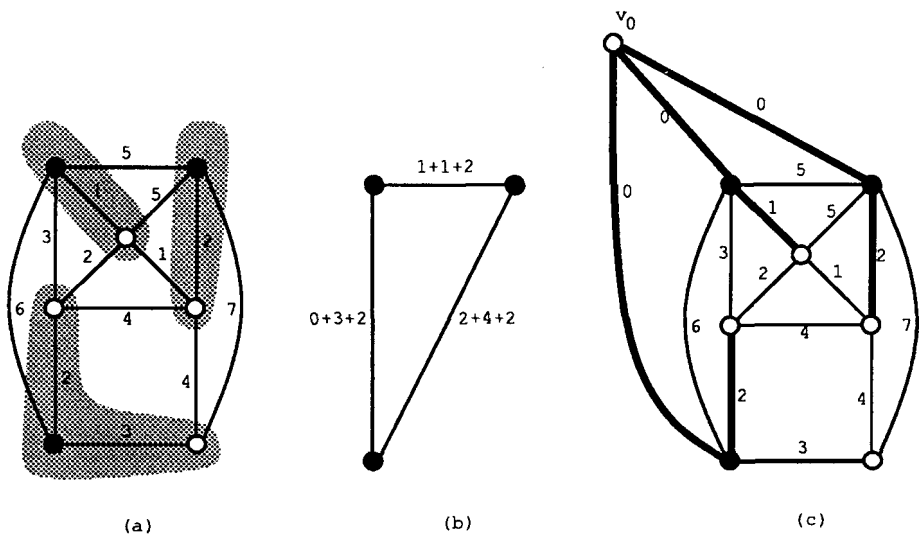

Figure 4.6: Neighborhoods of terminals

The neighborhoods $N(z_i)$, $z_i \in N$, can be found by a single shortest path computation. Add an artificial vertex v_0 to G, connect it by zero-length edges to all terminals. Find the shortest paths tree with v_0 as the root (Fig. 4.6c). When v_0 is deleted from this tree, each of the resulting components contains exactly one terminal; its non-terminals belong to the neighborhood of the terminal. All this requires $O(e + v \log v)$ time.

Theorem 4.3 ([22]) *Every minimum spanning tree of G' is also a minimum spanning tree of $D_G(N)$.*

A straightforward but technical proof is omitted. From Theorem 4.3 follows that Step 1 and Step 2 of the distance network heuristic can be replaced by the determination of a minimum spanning tree $T_{G'}(N)$. This requires $O(e + v \log v)$ time. This term still dominates the worst-case time complexity of the remaining

three steps. Floren [10] observed that when replacing the edges of $T_{G'}(N)$ by the corresponding paths in G, the resulting network will always be a tree. So in principle Step 4 and Step 5 can be avoided. However, if applied, these steps can result in a better solution. This is for example the case for the problem instance shown in Fig. 4.6.

4.2.4 Repetitive Distance Network Heuristic

Plesník [24] and independently Sullivan [31] suggested a modification of the distance network heuristic which in general yields better solutions but has higher worst-case time complexity bounds. For a given q, $0 \leq q \leq n-2$, form distance networks $D_G(Q \cup N)$ for each $Q \subseteq V \setminus N$ with $|Q| \leq q$. Use the distance network heuristic for each of these networks and take as T_{DNH} the one with minimum length. This repetitive heuristic is stronger than the distance network heuristic. As shown by Sullivan [31], the worst-case error ratio tends to $2 - q/(n-2)$ which is better than $2 - 2/n$ for $q \geq 2$. On the other hand, the worst-case time complexity of this heuristic is $O((v-n)^q n^2 + v^3)$. Note that for $q = n-2$, the heuristic is guaranteed to obtain a Steiner minimal tree. It is then identical to the enumeration algorithm described in Section 3.1.

4.2.5 Multiple Distance Network Heuristic

Diané and Plesník [7] suggested the following *multiple distance network heuristic* (MDNH). Let $T_{DNH}^{v_k}$ denote T_{DNH} for $N \cup v_k$, $v_k \notin N$. Let $T_{DNH}^{\leq v_k}$ denote T_{DNH} for $N \cup \{v_i \notin N \mid |T_{DNH}^{v_i}| \leq |T_{DNH}^{v_k}|\}$. Select as T_{MDNH} the shortest tree from among T_{DNH} and $T_{DNH}^{\leq v_k}$, $v_k \notin N$.

This heuristic performs at least as well as the distance network heuristic. Diané and Plesník [7] showed that $|T_{MDNH}|/|T_G(N)| \leq 2 - 2/n$. The worst-case time complexity of the heuristic is $O(ev + v^2 \log v)$.

4.2.6 Simulated Annealing Heuristics

The reader is assumed to be familiar with the general framework of the simulated annealing algorithm. It will be only briefly discussed below. The reader is referred to e.g., van Laarhoven and Aarts [33] for theoretical and practical aspects of this algorithm.

The simulated annealing algorithm starts with a feasible solution T at some initial *temperature* t. Another feasible solution T' in the *neighborhood* of T is selected. If T' is better than T, T' replaces T in the next iteration. Otherwise, T' replaces T with probability $e^{-(|T'|-|T|)/t}$. After a certain number $n(t)$ of iterations, the temperature is appropriately reduced. If the neighborhood structure is appropriately chosen (i.e., every feasible solution can reach any other feasible solution in a finite number of neighborhood transformations) and the cooling schedule reduces t at an appropriate rate (with the initial t sufficiently large), then it can be shown that the simulated annealing algorithm obtains a global optimum with probability

converging to 1 as the number of iterations goes to ∞. A *simulated annealing heuristic* (SAH) is derived from the algorithm by fixing the number of iterations.

Schiemangk [30] suggested a simulated annealing heuristic for the Steiner arborescence problem. Its counterpart for the Steiner tree problem is outlined below. A solution T is feasible if T is a tree spanning all terminals and where all non-terminals have degree at least 2. A neighbor of a feasible solution T is any tree T' that can be obtained from T by

- removing an edge e from T,
- reconnecting two components of $T - e$ by a path with minimum number of edges,
- removing non-terminals of degree 1 (one at a time).

Dowsland [9] suggested another simulated annealing heuristic for the Steiner tree problem. Feasible solutions are again trees spanning all terminals (with no non-terminals of degree 1). The neighborhood structure is however more elaborate. Current feasible solution T is broken into two subtrees T_i and T_j by removal of a randomly chosen elementary path (i.e., a path with end-vertices being either terminals or non-terminals of degree more than 2, and with intermediate vertices being non-terminals of degree 2). Three vertices are then selected at random: v_i belonging to T_i, v_j belonging to T_j and v_s belonging to non-terminals not in T extended by an artificial vertex v_0. If $v_s = v_0$, then T_i and T_j are reconnected by a shortest path from v_i to v_j. Otherwise, T_i and T_j are reconnected by shortest paths from v_i to v_s and from v_j to v_s. If these two shortest paths overlap before reaching v_s, only their disjoint parts are included in the new feasible solution. Dowsland [9] showed that the neighborhood structure defined in this way is sound (i.e., a Steiner minimal tree can be reached by a finite number of transformation from any feasible solution). In order to reduce the number of transformations, some of the reductions described in Chapter 2 are incorporated into the heuristic (i.e., some elementary paths are never chosen as they belong to at least one Steiner minimal tree, and some shortest paths are never used to interconnect subtrees as they do not belong to any Steiner minimal tree).

4.2.7 Hill-Climbing Heuristics

Dowsland [9] also suggested three *hill-climbing heuristics* (HCH). The first of them examines each elementary path in the current feasible solution T. Each elementary path is removed (one at a time) and the two subtrees are reconnected by a shortest possible path. T is replaced by the shortest tree found in this way. The heuristic stops when no improvements are possible.

The second hill-climbing heuristic examines all local transformations based on the sequential removal of two elementary paths followed by an appropriate reconnection scheme.

The third variant uses hybrid approach. The first hill-climbing heuristic is applied. Then one iteration of the second hill-climbing heuristic is applied. If this results in any improvement, the whole sequence is repeated.

4.3 Vertex Heuristics

The major difficulty when solving the Steiner tree problem is to identify non-terminals that belong to the Steiner minimal tree. Once given, the Steiner minimal tree can be found easily; it is a minimum spanning tree for the subnetwork induced by the terminals and selected non-terminals. The general idea behind vertex heuristics is to identify "good" non-terminals.

4.3.1 Average Distance Heuristic

The *average distance heuristic* (ADH) was suggested by Rayward-Smith [28]. It is based on the idea of connecting already constructed subtrees of a solution by shortest paths through some centrally located vertex. The degree of centrality of a vertex $v_i \in V$ is measured by some appropriately chosen measure $f(v_i)$ (to be defined below).

- **Step 1:** Begin with T_{ADH} consisting of the isolated terminals (Fig. 4.7a). Let $k = 1$.
- **Step 2:** If $k = n$, then **Stop**. Otherwise determine $f(v_i)$ for each $v_i \in V$, and select a vertex v_l with the smallest f-value.
- **Step 3:** Add the edges on the paths from v_l to the closest and second-closest components of T_{ADH} (Fig. 4.7b and Fig. 4.7c). $k := k + 1$. Go to **Step 2**.

Two additional steps, identical to Step 4 and Step 5 of the shortest paths heuristic, can be applied to further improve the solution.

One way of defining f during the k-th iteration, $1 \leq k \leq n-1$, is as follows [29]. Let $C_r^{v_i}$, $2 \leq r \leq n-k+1$, denote r trees in the partially constructed T_{ADH} that are closest to a vertex $v_i \in V$ (ties are broken arbitrarily). Define

$$f(v_i) = \min_r \{ \sum_{T_j \in C_r^{v_i}} d(v_i, T_j)/(r-1) \}$$

The worst-case time complexity of the average distance heuristic is dominated by the computation of shortest paths between all pair of vertices in G. Hence, its worst-case time complexity is $O(v^3)$ [11]. This bound can be reduced to $O(ve + v^2 \log v)$ for sparse networks.

Theorem 4.4 ([36]) *$|T_{ADH}|/|T_G(N)| \leq 2 - 2/l$ for any network G and any set N of terminals. Furthermore, for any ε, $\varepsilon > 0$ there exists a problem instance such that $|T_{ADH}|/|T_G(N)| > 2 - \varepsilon$.*

Proof. Let $T_D(N)$ denote the minimum spanning tree of $D_G(N)$. From Theorem 4.2 follows immediately that $|T_D(N)|/|T_G(N)| \leq 2 - 2/l$ where l is the number of leaves in $T_G(N)$. Waxman and Imase [36] proved that $|T_{ADH}| \leq |T_D(N)|$ (a tedious technical proof is omitted). The first part of the theorem follows.

In order to prove the second part, let k be any positive integer. Consider a full binary tree B_k of depth k with an additional path through the leaves (Fig. 4.8).

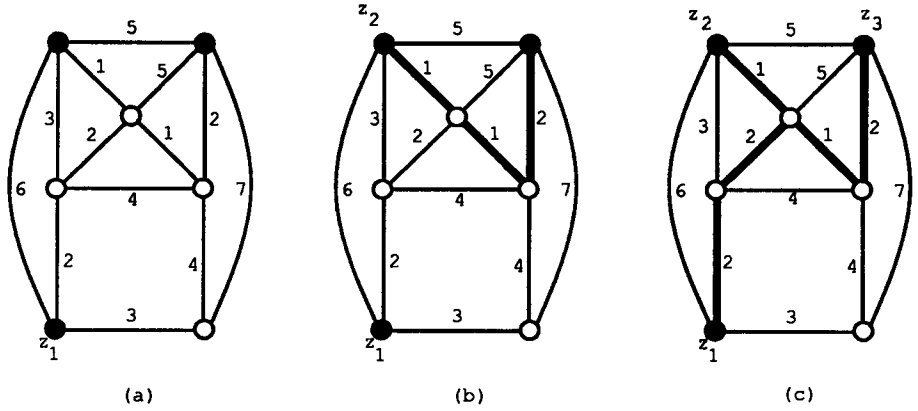

Figure 4.7: Average distance heuristic

Consider any non-leaf vertex v_i. Let h denote its height (number of edges which must be traversed to reach any leaf). Define the length of the two edges leaving v_i downward to be $2^{h-1} + \delta$, $0 < \delta \leq 1$. Define the length of the bottom edge connecting the two subtrees of v_i to be $2^{h+1} - 2$. Let N be the set of leaf-vertices.

For a fixed k, $k > 0$, T_{ADH} consists of the path through the leaves. Its length is $2(k2^k - 2^k + 1)$ On the other hand, $T_G(N)$ is the binary tree B_k. Its length is $k2^k + (2^{k+1} - 1)\delta$. Then

$$|T_{ADH}|/|T_G(N)| > \frac{2}{1 + 2\delta/k} - \frac{2}{k}$$

Given any $\varepsilon > 0$, then $|T_{ADH}|/|T_G(N)| > 2 - \varepsilon$ for any fixed δ, $0 < \delta < 1$, and sufficiently large k. □

Bern and Plassmann [4] analyzed the average distance heuristic for complete networks with edge lengths either 1 or 2. They proved that the worst-case error ratio is then tightly bounded by 4/3.

If $n = v$, the average distance heuristic reduces to the well-known minimum spanning tree algorithm of Kruskal [20]. If $n = 2$, the solution will be the minimum length path between the two terminals. Thus, in those two "boundary" cases, the average distance heuristic will yield a Steiner minimal tree.

4.3.2 Fast Average Distance Heuristic

The average distance heuristic connects two trees during each iteration. However, $f(v_l)$ can be attained for any subset $C_r^{v_l}$, $2 \leq r \leq n - k + 1$, of trees in the partially constructed T_{ADH}. It seems therefore natural to connect all trees in $C_r^{v_l}$ defining $f(v_l)$ to v_l via shortest paths. This usually reduces the number of iterations. Although this *fast average distance heuristic* (FADH) is less "cautious" than the

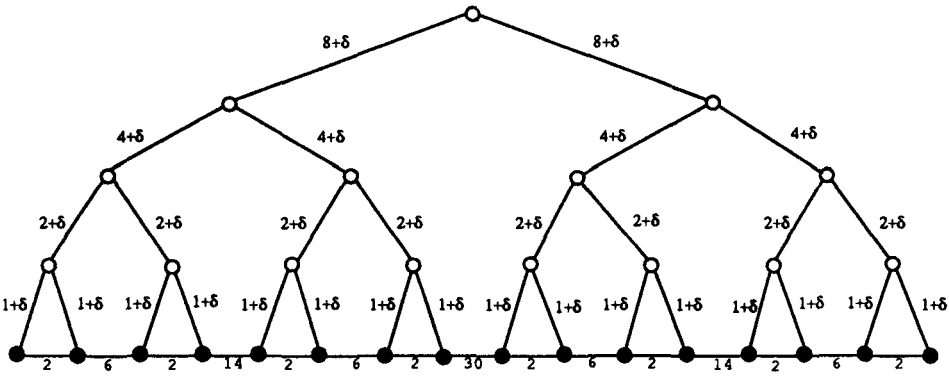

Figure 4.8: Tightness of the error ratio bound of T_{ADH}

average distance heuristic, Plesník [25] observed that this usually does not affect the quality of the solutions. He also proved that $|T_{FADH}|/|T_G(N)| \leq 2 - 2/n$ for any network G and any set N of terminals. Moreover, this bound is tight in the same sense as the bound for the error ratio of T_{ADH}. The worst-case time complexity of the fast average distance heuristic is $O(v^3)$.

4.3.3 Restricted Average Distance Heuristic

Chen [5] suggested a variant of the average distance heuristic that will be referred to as the *restricted average distance heuristic* (RADH). It joins three trees per iteration. Initially, T_{RADH} consists of n trees, each consisting of a terminal only. The selection is based on Steiner minimal trees for triplets of terminals. More specifically, let

$$f'(v_i) = \sum_{j=1}^{3} d(v_i, T_j^{v_i})$$

where $T_1^{v_i}$, $T_2^{v_i}$, $T_3^{v_i}$ are three subtrees of T_{RADH} closest to $v_i \in V$. A vertex v_l with the smallest f'-value is then selected, and the corresponding trees are joined together by a Steiner minimal tree. This is repeated as long as there are more than two subtrees left. If there are two subtrees left, they are joined by a shortest interconnecting path. The worst-case time complexity of the restricted average distance heuristic is $O(n^2 e \log v)$. Widmayer [38] showed that $|T_{RADH}|/|T_G(N)| \leq 2 - 2/n$. Plesník [25] showed that this bound is tight in the same sense as the bound for the error ratio of T_{ADH}.

4.3.4 Median Heuristic

Diané and Plesník [7] suggested the f ollowing *median heuristic* (MH). Determine

$$f''(v_i) = \sum_{z_k \in N} d(v_i, z_k)$$

for each $v_i \notin N$. Select v_l with the smallest f''-value. Consider the subnetwork of G induced by shortest paths from v_l to all terminals. Find a minimum spanning tree for this subnetwork. Remove (one at a time) non-terminals of degree one. Let $T_{DNH}^{v_l}$ denote the resulting tree. Let

$$T_{MH} = \begin{cases} T_{DNH}^{v_l} & \text{if } |T_{DNH}^{v_l}| < |T_{DNH}| \\ T_{DNH} & \text{otherwise} \end{cases}$$

Diané and Plesník [7] proved that $|T_{MH}|/|T_G(N)| \leq 2 - 2/n$. This bound is tight in the same sense as the bound for the error ratio of T_{ADH}. The worst-case time complexity of the median heuristic is $O((v-n)v^2)$.

4.3.5 Antimedian Heuristic

Diané and Plesník [7] suggested also the following *antimedian heuristic* (AMH). As in the median heuristic, determine $f''(v_i)$ for every $v_i \notin N$. Remove from G (one at a time) non-terminals in non-increasing order of their f''-values provided that the resulting network contains all terminals in the same component. Upon completion, determine a minimum spanning tree of the component containing all terminals. Delete from this tree (one at a time) non-terminals of degree one. Let T_{AMH}^* denote the resulting tree. Let

$$T_{AMH} = \begin{cases} T_{AMH}^* & \text{if } |T_{AMH}^*| < |T_{DNH}| \\ T_{DNH} & \text{otherwise} \end{cases}$$

Diané and Plesník [7] proved that $|T_{AMH}|/|T_G(N)| \leq 2 - 2/n$. This bound is tight in the same sense as the bound for the error ratio of T_{ADH}. The worst-case time complexity of the antimedian heuristic is $O((v-n)v^2)$.

4.3.6 Upgrading Heuristic

Zelikovsky [42] suggested a vertex heuristic that is very important from the theoretical point of view. It is the first polynomial heuristic with the worst-case error ratio bounded by a constant less than 2. Furthermore, it can be generalized to a heuristic for the Steiner problem in an arbitrary metric space (see Part IV, Section 1.4). This is in particular the case for the Euclidean metric (see Part I, Section 4.8) and for the rectilinear metric (see Part III, Subsection 2.2.6).

The main idea behind the heuristic is to identify some non-terminals, include them into the set of terminals, and apply the distance network heuristic to this extended set of terminals. This heuristic will be referred to as the *upgrading heuristic* (UH) since some non-terminals are "upgraded" to terminals. The upgrading heuristic is as follows.

- **Step 1:** $W = \emptyset$. $w = \infty$. $D = D_G(N)$.
- **Step 2:** Identify a set of three terminals $N_i = \{z_i, z_j, z_k\}$ in D such that

$$w = |T_D(N)| - |T_D(\bar{N})| - |T_G(N_i)|$$

is maximized and where $\bar{D}$ denotes a network obtained from D by contraction of the three edges (z_i, z_j), (z_j, z_k), (z_k, z_i) while $\bar{N}$ denotes the vertices of $\bar{D}$.

- **Step 3:** If $w = 0$, then let T_{UH} be T_{DNH} for $N \cup W$ and **Stop.** Otherwise add to W the non-terminal v_p of degree 3 in $T_G(N_i)$ (it always exists). Let $D = \bar{D}$, and go to **Step 2.**

Theorem 4.5 ([42]) $|T_{UH}|/|T_G(N)| \leq 11/6$ *for any network* G *and any set* N *of terminals.*

The complicated proof of this theorem requiring several technical lemmas is omitted. The worst-case time complexity of the heuristic is $O(v^3 + vn^3)$.

4.3.7 Modified Upgrading Heuristic

During each iteration of the upgrading heuristic one non-terminal is identified and added to the set of terminals for which T_{DNH} is determined in the final stage. The inclusion of a non-terminal is irreversible and may therefore cause omission of some other non-terminals whose impact on the final solution would be even greater. Furthermore, the identification of non-terminals upgraded to terminals is based on the determination of Steiner minimal trees for triples of original terminals in G.

Berman and Ramaiyer [2,3] suggested a *modified upgrading heuristic* (MUH). Non-terminals are not upgraded to terminals immediately. Instead, they are pushed on a stack. Only after Steiner minimal trees for all small (not only triplets) subsets of original terminals in G have been considered, final decision which non-terminals should be upgraded is made.

Let N_i denote a subset of terminals, $3 \leq |N_i| \leq k$. Let $E_i = \{(z_j, z_l) | z_j, z_l \in N_i\}$. Let

$$w = |T_D(N)| - |T_{\bar{D}}(\bar{N})| - |T_G(N_i)|$$

where $D = D_G(N)$ and $\bar{D}$ is obtained by contracting D along $|N_i| - 1$ edges of E_i. If $w > 0$, then push $T_G(N)$ and E_i on the stack. With each edge $(z_j, z_l) \in E_i$ is associated a modified length $d(z_j, z_l) - w$. Apart from pushing E_i on the stack, it is also added to D (whereby parallel edges may occur). This process is repeated for all subsets with up to k terminals.

In the second phase of the modified upgrading heuristic, pairs of Steiner minimal trees and edge sets are removed from the stack one by one. Suppose that some $T_G(N_i)$ and E_i is removed from the stack. If $T_D(N) \cap E_i$ contains $|N_i| - 1$ edges, all non-terminals of $T_G(N_i)$ are upgraded to terminals. Otherwise E_i is removed from D.

When the stack becomes empty, T_{DNH} for N and the upgraded non-terminals in G is determined and returned as T_{MUH}.

If k is fixed, Steiner minimal trees for subsets with up to k terminals can be determined in polynomial time (using e.g., the exact dynamic programming algorithm discussed in Section 3.4). Since the number of subsets is also polynomial, the modified upgrading heuristic is polynomial. A detailed time complexity analysis is omitted as the modified upgrading heuristic cannot compete in this respect with other heuristics. The importance of the modified upgrading heuristic is due to the following theorem (the complicated technical proof is omitted).

Theorem 4.6 ([2,3]) $|T_{MUH}|/|T_G(N)| \leq 16/9$ *for* $k = 4$ *and for any network* G *and any set* N *of terminals.*

4.4 Contraction Heuristic

As it was pointed out in Subsection 2.2.2, if a network G contains a terminal z_i, and the shortest edge incident to it has a terminal z_j as the other endpoint, then (z_i, z_j) belongs to the Steiner minimal tree. It then suffices to consider the Steiner tree problem arising after the contraction of G along (z_i, z_j). The *contraction heuristic* (CH) given by Plesník [24] is based on a more general idea of contraction.

In order to describe the contraction heuristic, some definitions are needed. Let $z_i \in N$ and $r > 0$ be given. A neighborhood $N_r(z_i)$ of z_i with radius r is the set $\{x \in G \mid d_G(x, z_i) \leq r\}$. Note that x does not need to be a vertex belonging to V, but any point on the edges (regarded as simple curves) of G. Two neighborhoods $N_r(z_i)$ and $N_r(z_j)$, $z_i, z_j \in N$, are said to be reachable from one another if they are adjacent or there is a path from z_i to z_j entirely within a union of neighborhoods of some terminals. A *neighborhood class* C_p is the maximal union of neighborhoods that are reachable from one another. Classes arising in connection with the network shown in Fig. 4.9 (for $r = 1$) are indicated by shaded regions.

A class contraction of a network G that produces a network $\bar{G}$ is obtained in the following way.

- Each class C_p is contracted to a single vertex z_p regarded as a terminal in $\bar{G}$. Non-terminals not belonging to any class are left unchanged in $\bar{G}$.
- An edge $e_m = (v_i, v_j)$ in G with end-vertices not belonging to the same class generates an edge e_m in $\bar{G}$ according to the following rules (Fig. 4.9b):
  - If neither v_i nor v_j belongs to any class, then the edge e_m remains unchanged in $\bar{G}$.
  - If v_i belongs to a class C_p and v_j belongs to no class (i.e., $v_j \notin N$), then $\bar{G}$ contains the edge $e_m = (z_p, v_j)$ of length

$$c(z_p, v_j) = \begin{cases} c(v_i, v_j) - r & \text{if } v_i \in N \\ c(v_i, v_j) & \text{if } v_i \notin N \end{cases}$$

  - If v_i belongs to a class C_p and v_j to another class C_q, then $\bar{G}$ contains the edge $e_m = (z_p, z_q)$ of length

$$c(z_p, z_q) = \begin{cases} c(v_i, v_j) - 2r & \text{if } v_i, v_j \in N \\ c(v_i, v_j) & \text{if } v_i, v_j \notin N \\ c(v_i, v_j) - r & \text{otherwise} \end{cases}$$

- All but the minimum length edge connecting any pair of vertices in $\bar{G}$ are deleted (Fig. 4.9c).

The recursive heuristic based on the class contractions is as follows.

- **Step 1:** Determine the minimum length edge in G incident to a terminal. Let r denote its length. Form neighborhoods $N_r(z_i)$, $z_i \in N$, and the corresponding classes C_p, $p = 1, 2, ..., t$.

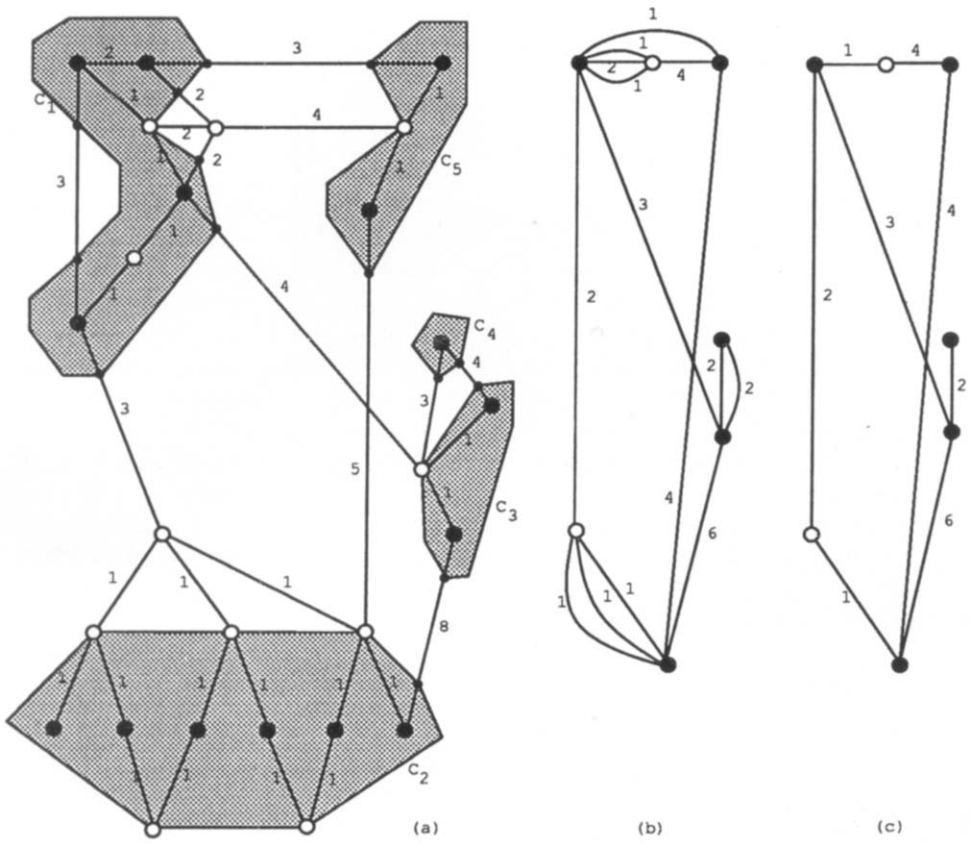

Figure 4.9: Contraction heuristic

- **Step 2:** For each class C_p, $p = 1, 2, ..., t$, let N_p denote the terminals in C_p. Note that for any terminal in N_p, there is another terminal in N_p (unless $|N_p| = 1$) not farther away than $2r$. Thus a tree T_p spanning N_p with $|T_p| \leq 2r(|N_p| - 1)$ can be determined by the shortest paths heuristic (Subsection 4.1.1).
- **Step 3:** If $t = 1$, then T_1 is a tree spanning N in G; **Return.** Otherwise, let $\bar{G}$ be the network obtained from G by class contraction. Let $\bar{N}$ denote the terminals of $\bar{G}$.
- **Step 4:** Determine a tree $\bar{T}$ spanning $\bar{N}$ in $\bar{G}$ (by recursion).
- **Step 5:** Replace each $z_p \in \bar{N}$ by T_p for $p = 1, 2, ..., t$. Reconnect $\bar{T}$ and $T_1, T_2, ..., T_t$ by adding at most $\sum_{z_p \in \bar{N}} deg_G(z_p)$ edges, each of length r; the nature of class contractions makes such reconnection always possible. The resulting tree T_{CH} is the solution. **Stop.**

None of the steps in the heuristic requires more than $O(v^2)$ time. Since at most v recursions are needed, it follows that its worst-case time complexity is $O(v^3)$.

Theorem 4.7 ([24]) *$|T_{CH}|/|T_G(N)| \leq 2 - 2/n$ for any network G and any set of*

terminals N. Furthermore, this bound is tight, i.e., for any ε, $\varepsilon > 0$, there is an instance of the Steiner tree problem such that $|T_{CH}|/|T_G(N)| > 2 - \varepsilon$.

Plesník [26] analyzed the contraction heuristic in more detail. In particular, he provided a more precise worst-case error ratio depending on additional parameters. He also suggested a modification of the contraction heuristic where Step 2 and Step 5 are united. While this modification has the same worst-case time complexity and error ratio, it sometimes yields better solutions.

4.5 Dual Ascent Heuristic

The dual ascent method for the determination of a lower bound for the Steiner arborescence problem was described in Section 3.7. Wong [40] suggested how this method can be used to find a suboptimal solution. The *dual ascent heuristic* (DAH) is as follows.

- **Step 1:** Determine the directed network H as described in Section 3.7. Disregard the orientation of arcs. Let Q denote the set of vertices in H connected by a path with the terminal z_1. Note that $N \subseteq Q$.
- **Step 2:** Determine a minimum spanning tree of the subnetwork of G induced by Q. Denote it by T_{DAH}.
- **Step 3:** Remove from T_{DAH} non-terminals of degree 1 (one at a time) and **Stop**.

Wong [40] did not analyze this heuristic in terms of the worst-case time complexity. Furthermore, nothing is known about the worst-case error-ratio of T_{DAH}. However, experimental results (see Section 4.7) indicate that the dual ascent heuristic yields very good suboptimal solutions.

4.6 Set Covering Heuristic

The set covering algorithm for the Steiner tree problem was described in Section 3.10. Aneja [1] suggested to use this method to obtain a good feasible solution. The *set covering heuristic* (SCH) is as follows.

- **Step 1:** Determine the optimal solution $X = (x_{ij} | [v_i, v_j] \in A)$ to the linear relaxation of the set covering formulation $\mathcal{P}_4$. Let T_{SCH} denote a subnetwork of G containing every arc $[v_i, v_j]$ with $x_{ij} > 0$.
- **Step 2:** Let $[v_i, v_j]$ denote an arc of T_{SCH} such that $T_{SCH} - [v_i, v_j]$ is connected and x_{ij} is as small as possible. If no such $[v_i, v_j]$ exists, then **Stop**.
- **Step 3:** Let $T_{SCH} = T_{SCH} - [v_i, v_j]$ and go to **Step 2**.

No worst-case time complexity bound for this heuristic is available. Nothing is known about the worst-case error ratio of T_{SCH}. Limited computational results do not indicate that the set covering heuristic yields exceptionally good suboptimal solutions.

4.7 Summary and Computational Experience

Table 4.1 gives an overview of the heuristics for the Steiner tree problem discussed in this chapter.

Comparative analysis of most heuristics described in this chapter carried out by Rayward-Smith [28] and Plesník [25,26] shows that there is no clear dominance relation between them. More specifically, Plesník [25,26] examined the following 8 heuristics: shortest paths heuristic (SPH), Kruskal-based heuristic (KBH), Y-heuristic (YH), distance network heuristic (DNH), average distance heuristic (ADH), fast average distance heuristic (FADH), restricted average distance heuristic (RADH), and contraction heuristic (CH). For any pair H_a and H_b of these heuristics (except DNH versus CH), he constructed problem instances where H_a beats H_b and problem instances where H_b beats H_a.

Rayward-Smith and Clare [29] compared the shortest paths heuristic (SPH), distance network heuristic (DNH), and average distance heuristic (ADH). Distance network heuristic turned out to be less accurate than both shortest paths and average distance heuristics for various kinds of networks. On average, the average distance heuristic performed better than the shortest paths heuristic, although there were cases where this was not the case. However, the differences between lengths of obtained solutions were not large. Similar pattern of behavior of these three heuristics was observed by Voss [34] and Winter and Smith [39].

The error ratio for all three heuristics was far below the worst-case error ratio $2 - 2/n$. However, this was observed only for very sparse networks. For larger and denser networks, the optimal solutions were not available.

The repetitive variants of the shortest paths heuristic were compared by Winter and Smith [39]. In particular, the SPH-V and SPH-NN variants turned out to perform very well. They outperformed the average distance heuristic for all kinds of tested networks. This in particular indicates that there might be some improvements available in connection with the average distance heuristic if a more suitable centrality measure f is used.

Voss [34] carried out an extensive comparison which in addition to the shortest paths heuristic (SPH), distance network heuristic (DNH) and average distance heuristic (ADH) also included Y-heuristic (YH), restricted average distance heuristic (RADH), repetitive shortest paths heuristic (SPH-zN) and dual ascent heuristic (DAH). In particular, the dual ascent heuristic seemed to perform better than other heuristics. The only exception occurred in connection with the grid networks (vertical and horizontal lines through terminals). For such networks, the repetitive shortest paths heuristic (SPH-zN) performed better. It remains to be seen whether the most efficient repetitive shortest paths heuristics (SPH-V and SPH-NN) can compete with the dual ascent heuristic on other types of networks. They performed better than the SPH-zN on grid networks [39].

Simulated annealing (SAH) and hill-climbing (HCH) heuristics seem to be poor alternatives. They typically require good starting solutions. Unless elaborate local exchanges are used, these heuristics require many iterations. Furthermore, in the case of hill-climbing heuristics, the quality of obtained solution is usually poor.

Method	Submethod	Acronym	Complexity	Error Ratio	Reference	Section
Path heuristics	shortest paths	SPH	$O(n(e + v \log v))$	$2 - 2/n$	Takahashi and Matsuyama [32]	4.1.1
	repetitive	SPH-N	$O(n^2v^2)$	$2 - 2/n$	Winter and Smith [39]	4.1.2
		SPH-V	$O(nv^3)$			
		SPH-zN	$O(n^2v^2)$			
		SPH-NN	$O(n^3v^2)$			
	with origin	SPOH	$O(e + v \log v)$	$n - 1$	Takahashi and Matsuyama [32]	4.1.3
	Kruskal	KBH	$O(nv^2)$	$2 - 2/n$	Wang [35]	4.1.4
	Y	YH	$O(n^2e \log v)$	$2 - 2/n$	Chen [5]	4.1.5
Tree heuristics	min. spanning trees	MSTH	$O(e + v \log v)$	$v - n + 1$	Takahashi and Matsuyama [32]	4.2.1
	greedy	GTH	$O((v-n)^2v + n^2)$	$v - n + 1$	Minoux [23]	4.2.2
	distance network	DNH	$O(e + v \log v)$	$2 - 2/n$	Mehlhorn [22] and [6,16,17,19,24,35,41]	4.2.3
	repetitive	RDNH	$O((v-n)^q n^2 + v^3)$	$2 - q/(n-2)$	Plesník [24] Sullivan [31]	4.2.4
	multiple	MDNH	$O(ev + v^2 \log v)$	$2 - 2/n$	Diané and Plesník [7]	4.2.5
	simulated annealing	SAH			Schiemangk [30] Dowsland [9]	4.2.6
	hill-climbing	HCH			Dowsland [9]	4.2.7
Vertex heuristics	average distance	ADH	$O(v^3)$	$2 - 2/n$	Rayward-Smith and Clare [28,29]	4.3.1
	fast	FADH	$O(v^3)$	$2 - 2/n$	Plesník [26]	4.3.2
	restricted	RADH	$O(n^2e \log v)$	$2 - 2/n$	Chen [5]	4.3.3
	median	MH	$O((v-n)v^2)$	$2 - 2/n$	Diané and Plesník [7]	4.3.4
	antimedian	AMH	$O((v-n)v^2)$	$2 - 2/n$	Diané and Plesník [7]	4.3.5
	upgrading	UH	$O(v^3 + vn^3)$	11/6	Zelikovsky [42]	4.3.6
	modified upgrading	MUH		16/9	Berman and Ramaiyer [2,3]	4.3.7
Contraction		CH	$O(v^3)$	$2 - 2/n$	Plesník [24,26]	4.4
Dual Ascent		DAH			Wong [40]	4.5
Set Covering		SCH			Aneja [1]	4.6

Table 4.1: Heuristics for the Steiner tree problem

No computational experience for the upgrading heuristic (UH) and modified upgrading heuristic (MUH) is currently available. Although these heuristics have worst-case error bounds below 2, it is uncertain whether they can produce solutions that on average are better than the solutions obtained by the best repetitive shortest paths heuristics or by the dual ascent heuristic. Furthermore, upgrading heuristics are far from efficient as their computational load is very high.

References

[1] Y. P. Aneja, An integer linear programming approach to the Steiner problem in graphs *Networks* **10** (1980) 167-178.

[2] P. Berman and V. Ramaiyer, An approximation algorithm for the Steiner tree problem, Technical Report CS-91-05, The Pennsylvania State University (1991).

[3] P. Berman and V. Ramaiyer, Improved approximations for the Steiner tree problem, Technical Report CS-91-10, The Pennsylvania State University (1991).

[4] M. W. Bern and P. Plassmann, The Steiner problem with edge lengths 1 and 2, *Inf. Process. Lett.* **32** (1989) 171-176.

[5] N. P. Chen, New algorithm for Steiner tree on graphs, *IEEE Symp. on Circuits and Systems* (1983) 1217-1219.

[6] E.-A. Choukhmane, Une heuristique pour le probleme de l'arbre de Steiner, *RAIRO Rech. Opér.* **12** (1978) 207-212.

[7] M. Diané and J. Plesník, Three new heuristics for the Steiner problem in graphs, *Acta Math. Univ. Comenianae* **60** (1991) 105-121.

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

[9] K. A. Dowsland, Hill-climbing, simulated annealing and the Steiner problem in graphs, *Eng. Optimization* **17** (1991) 91-107.

[10] R. Floren, A note on "A faster approximation algorithm for the Steiner problem in graphs", *Inf. Process. Lett.* **38** (1991) 177-178.

[11] R. W. Floyd, Algorithm 97: Shortest path, *Comm. ACM* **5** (1962) 344-345.

[12] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. Assoc. Comput. Mach.* **34** (1987) 596-615.

[13] H. N. Gabow, Z. Galil and T. H. Spencer, Efficient implementation of graph algorithms using contraction, *J. Assoc. Comput. Mach.* **36** (1989) 540-572.

[14] M. X. Goemans, Survivable networks and parsimonious property, Technical Report, Oper. Res. Center, MIT (1990).

[15] M. X. Goemans and D. J. Bertsimas, On the parsimonious property of connectivity problems, Technical Report, Oper. Res. Center, MIT (1989).

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

[17] L. T. Kou, On efficient implementation of an approximation algorithm for the Steiner tree problem, *Acta Inf.* **27** (1990) 369-380.

[18] L. T. Kou and K. Makki, An even faster approximation algorithm for the Steiner tree problem in graphs, *Congr. Numerantium* **59** (1987) 147-154.

[19] L. T. Kou, G. Markowsky and L. Berman, A fast algorithm for Steiner trees, *Acta Inf.* **15** (1981) 141-145.

[20] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. Am. Math. Soc.* **7** (1956) 48-50.

[21] L. Kucera, A. Marchetti-Spaccamela, M. Protasi and M. Talamo, Near optimal algorithms for finding minimum Steiner trees on random graphs, in J. Gruska (ed.) *Mathematical Foundations of Computer Science*, **LNCS 233**, Springer-Verlag (1986) 501-511.

[22] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs, *Inf. Process. Lett.* **27** (1988) 125-128.

[23] M. Minoux, Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity, *INFOR* **28** (1990) 221-233.

[24] J. Plesník, A bound for the Steiner problem in graphs, *Math. Slovaca* **31** (1981) 155-163.

[25] J. Plesník, Worst-case relative performances of heuristics for the Steiner problem in graphs, *Acta Math. Univ. Comenianae* **60** (1991) 269-284.

[26] J. Plesník, On heuristics for the Steiner problem in graphs (to appear).

[27] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Tech. J.* **36** (1957) 1389-1401.

[28] V. J. Rayward-Smith, The computation of nearly minimal Steiner trees in graphs, *Int. J. Math. Educ. Sci. Technol.* **14** (1983) 15-23.

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

[30] C. Schiemangk, Thermodynamically motivated simulation for solving the Steiner tree problem and the optimization of interacting path systems, in A. Iwainsky (ed.), *Optimization of Connection Structures in Graphs* , CICIP, East Berlin, GDR (1985) 74-90.

[31] G. F. Sullivan, Approximation algorithms for Steiner tree problems, Tech. Report 249, Dept. of Computer Science, Yale Univ. (1982).

[32] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Jap.* **24** (1980) 573-577.

[33] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel Publishing Company, Dordrecht, Holland (1987).

[34] S. Voss, Steiner's problem in graphs: Heuristic methods, to appear in *Discrete Appl. Math.*

[35] S.M. Wang, A multiple source algorithm for suboptimum Steiner trees in graphs, in H. Noltemeier (ed.) *Proc. Int. Workshop on Graph-Theoretic Concepts in Computer Science*, Würzburg (1985) 387-396.

[36] B. M. Waxman and M. Imase, Worst-case performance of Rayward-Smith's Steiner tree heuristics, *Inf. Process. Lett.* **29** (1988) 283-287.

[37] P. Widmayer, On approximation algorithms for Steiner's problem in graphs, in G. Tinhofer and G. Schmidt (eds.), *Graph-Theoretic Concepts in Computer Science*, **LNCS 246**, Springer-Verlag (1986) 17-28.

[38] P. Widmayer, Fast approximation algorithms for Steiner's problem in graphs, (Habilitation Thesis, 1986), Inst. für Angewandte Informatik und Formale Beschreibungsverfahren, Univ. Karlsruhe (1987).

[39] P. Winter and J. MacGregor Smith, Path-distance heuristics for the Steiner problem in undirected networks, *Algorithmica* **7** (1992) 309-327.

[40] R. T. Wong, A dual ascent approach for the Steiner tree problem on a directed graph, *Math. Program.* **28** (1984) 271–287.

[41] Y. F. Wu, P. Widmayer and C. K. Wong, A faster approximation algorithm for the Steiner problem in graphs, *Acta Inf.* **23** (1986) 223-229.

[42] A. Z. Zelikovsky, An 11/6-approximation algorithm for the Steiner problem on networks, to appear in *Proc. of the 4-th Czechoslovak Symp. on Combinatorics.*