Algorithmica (1992) 7:309–327

Algorithmica

Path-Distance Heuristics for the Steiner Problem in Undirected Networks

Pawel Winter[1] and J. MacGregor Smith[2]

Abstract. An integrative overview of the algorithmic characteristics of three well-known polynomial-time heuristics for the undirected Steiner minimum tree problem: *shortest path heuristic* (SPH), *distance network heuristic* (DNH), and *average distance heuristic* (ADH) is given. The performance of these *single-pass* heuristics (and some variants) is compared and contrasted with several heuristics based on *repetitive* applications of the SPH. It is shown that two of these repetitive SPH variants generate solutions that in general are better than solutions obtained by any single-pass heuristic. The worst-case time complexity of the two new variants is $O(pn^3)$ and $O(p^3n^2)$, while the worst-case time complexity of the SPH, DNH, and ADH is respectively $O(pn^2)$, $O(m + n \log n)$, and $O(n^3)$ where p is the number of vertices to be spanned, n is the total number of vertices, and m is the total number of edges. However, use of few simple tests is shown to provide large reductions of problem instances (both in terms of vertices and in term of edges). As a consequence, a substantial speed-up is obtained so that the repetitive variants are also competitive with respect to running times.

Key Words. Steiner problem, Undirected graphs, Heuristics, Complexity.

1. Introduction. The *Steiner minimum tree* (SMT) problem is one of the most well-known combinatorial optimization problems. It asks for the minimum total subnetwork spanning some of the vertices of an undirected network. The decision version of the SMT problem, contrary to the *minimum spanning tree* (MST) problem, where all vertices have to be spanned, is known to be NP-complete [9]. It is therefore very unlikely that a polynomial algorithm for the SMT problem exists. Several exponential enumeration algorithms have been suggested [23]. Due to the inherent difficulty in solving the SMT problem, much effort has been devoted to the development of polynomial heuristics which produce good quality solutions.

We have a threefold purpose in this paper. First, we characterize and provide a common framework for three well-known heuristics for the SMT problem: *shortest path heuristic* (SPH), *distance network heuristic* (DNH), and *average distance heuristic* (ADH). We also discuss some variants of the SPH. We argue that these heuristics depend to a large extent on distance calculations and iterative enlargement of partial solutions by addition of shortest paths between appropriately chosen vertices. We refer to this class of heuristics as *single-pass path-distance heuristics* (PDH). The interrelationship between the SPH, DNH, and ADH was

[1] Institute of Datalogy, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark.

[2] Department of Industrial Engineering and Operations Research, University of Massachusetts, Amherst, MA 01003, USA.

Received October 15, 1989; revised March 29, 1990. Communicated by F. K. Hwang.

previously recognized in a slightly different form in [17]. Second, we extend the class PDH by various known and new heuristics based on the repetitive applications of the SPH, and on the use of problem reduction tests. Third, we present a comprehensive computational analysis of the PDHs so that a concrete understanding of their performance is developed. In particular, we argue that two of the repetitive heuristics on average produce solutions which are better than those obtained by any single-pass heuristic. Furthermore, the reductions make their computational times competitive with the computational times of single-pass heuristics.

1.1. Problem Formulation. The SMT problem in an undirected network $G = (V, E, c)$ is defined as follows:

- *Given*: A connected undirected network $G = (V, E, c)$ with n vertices and m edges, and a subset $Z \subseteq V$ of p vertices.
- *Find*: A minimum cost connected subnetwork T of G which spans all Z-vertices.

If the edge-cost function c is positive, the solution to the SMT problem is bound to be a tree. If $Z = V$, then the problem becomes that of finding the MST in G. If $p = 2$, then the problem reduces to the shortest path problem between the two Z-vertices. In the following we therefore assume that Z is a proper subset of V consisting of at least three vertices.

Figure 1 illustrates a morphological tree structure of the various types of approaches to the SMT problem along with some of the most relevant literature citations. We focus on the PDH class, as we wish to integrate and extend this class of heuristics as well as provide a detailed analysis of their relative performance.

1.2. Path-Distance Heuristics (PDH). The PDHs can be considered as derivatives from the exact approaches to the SMT problem as well as from the algorithms for the MST problem. The search for an optimal solution to the SMT problem amounts to the determination of its S-vertices where $S = V \backslash Z$. The search for an optimal solution to the MST problem amounts to the determination of appropriate edges. Consequently, it seems reasonable to base heuristic approaches to the SMT problem on iterative enlargements of partial solutions by appropriately chosen shortest paths (possibly through some prespecified intermediate vertices). Path selection corresponds to edge selection in MST algorithms. S-vertices occurring on these paths correspond to S-vertices selected by exact algorithms for the SMT problem.

Heuristics in the PDH class have the following general structure:

- *Step 1: Initialization.* Begin with the initial partial solution $T = (Z, \varnothing)$ being a forest consisting of isolated Z-vertices.
- *Step 2: Path Selection.* Add to T an appropriately chosen path connecting two components of T.
- *Step 3: Termination Test.* Repeat step 2 until T becomes connected.

Steps 1 and 3 remain unchanged for all single-pass PDHs. It is step 2 which is

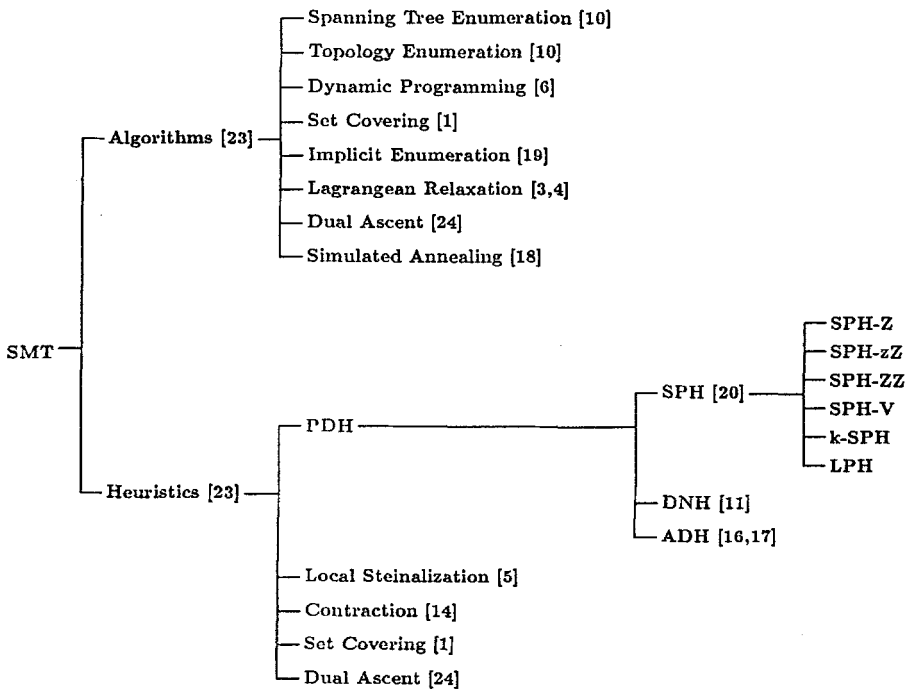

Fig. 1. Morphological structure of approaches to the SMT problem.

responsible for variations among PDHs. Path selection is based on shortest path information between all vertices. Such information can be obtained as needed during the iterative process or beforehand during the initialization phase. The latter alternative was used in the implementations and is assumed in the following.

As noted by Rayward-Smith and Clare [17], when the above heuristic terminates, T spans all Z-vertices and a subset S_T (possibly empty) of S-vertices. It follows immediately that the following two steps can further improve the solution:

- *Step 4: MST Computation.* Let T be the MST of the subnetwork of G induced by $Z \cup S_T$.
- *Step 5: Trimming.* Delete from T all S-vertices of degree 1 (one at a time). The resulting tree is the (suboptimal) solution. STOP.

All heuristics described in this paper employ steps 4 and 5.

1.3. Overview of the Paper. Section 2 of the paper describes the SPH and some of its modifications. In particular, various schemes for repetitive applications of the SPH (where not one but several trees spanning Z are generated) are discussed. Section 3 describes the DNH. Section 4 describes the ADH. Section 5 discusses tests which can be applied to reduce the size of the original problem instance. These tests are so effective that modifications based on the repetitive applications of the SPH become competitive (in terms of execution times) with other PDHs.

Section 6 describes the experimental design and computational comparison of all heuristics in the PDH class. Finally, Section 7 summarizes and concludes the paper.

2. Shortest Path Heuristic (SPH). The SPH was originally developed by Takahashi and Matsuyama [20]. Its step 2 is as follows:

- *Step 2.* Determine a Z-vertex z' closest to a subtree of T containing an *a priori* chosen Z-vertex z (ties are broken arbitrarily). Add to T the shortest path joining z' with the subtree containing z.

It is easily seen that the SPH is related to Prim's algorithm for the MST problem [15]. The SPH grows only one subtree which is expanded during each iteration by adding the closest Z-vertex z' together with the edges and S-vertices on the shortest path from z' to the subtree.

The worst-case time complexity of the SPH is $O(pn^2)$. As shown in [20] the worst-case error ratio between the solution T obtained by the SPH and the optimal solution T_0 is never greater than 2. More specifically,

$$\frac{c(T)}{c(T_0)} \leq 2\left(1 - \frac{1}{p}\right).$$

Furthermore, this bound is tight in the sense that there are problem instances for which the ratio equals the bound.

2.1. Longest Path Heuristic (LPH). The following modification of the SPH, which we refer to as the LPH, is an obvious and novel antithesis of the SPH. Rather than selecting the fixed Z-vertex z arbitrarily, and then connecting it during the first iteration to the closest Z-vertex z', identify the *longest* among shortest paths between *all* pairs of Z-vertices. Let the initial T consist of this path and of all remaining isolated Z-vertices. Determine a suboptimal solution by applying the SPH from that point.

The rationale behind the LPH is as follows. There must be a path between every pair of Z-vertices. By always selecting the closest Z-vertex as is done in the SPH, the resulting tree may have some paths between Z-vertices considerably longer than the shortest paths in the underlying network. In particular, it is reasonable to assume that Z-vertices farthest apart in G will have the greatest impact on the total cost of suboptimal solutions. Thus such Z-vertices should be connected by as short a path as possible.

On the other hand, the LPH might appear hard to justify since longest of shortest paths can consist of a single long edge (z, z') while there can be just slightly longer paths between z and z' which involve S-vertices. Inclusion of these S-vertices can reduce the total cost of paths subsequently added to the solution.

It is possible to construct problem instances where the LPH behaves better than the SPH. However, as will be seen in Section 6, the SPH in general performs much better than the LPH.

In conclusion, although the LPH cannot be considered as a serious competitor to the SPH, its surprisingly good behavior for some problem instances motivated our interest in repetitive heuristics discussed in the next subsection.

The worst-case time complexity of the LPH is the same as the worst-case time complexity of the SPH. However, it remains an open problem whether the worst-case error ratio is bounded by 2. Note that problem instances of arbitrary size with the worst-case error ratio asymptotically approaching 2 can be easily constructed. Consider for example a complete network of n Z-vertices with all edge-costs equal to 2. Add an additional S-vertex connected to all Z-vertices by edges of cost $1 + \varepsilon$. The solution obtained by the LPH will avoid the S-vertex and will have total cost equal to $2(n - 2)$. On the other hand, the optimal solution will contain the S-vertex and will have total cost equal to $(1 + \varepsilon)(n - 1)$.

2.2. Repetitive Shortest Path Heuristics. A straightforward improvement of the SPH is to run it for every possible choice of the fixed Z-vertex, and then take the best solution found. We refer to this heuristic as the SPH-Z. Its worst-case time complexity is $O(p^2n^2)$.

Further generalization of the SPH-Z is to repeat the SPH n times, once for each Z- and S-vertex (rather than once for each Z-vertex). It is natural to denote this heuristic SPH-V. The rationale behind this extension stems from the fact that the SPH and SPH-Z may fail to recognize S-vertices not on shortest paths but which nevertheless are useful in connection with the construction of low-cost Steiner trees. Such S-vertices will be taken into account by the SPH-V.

It should be noted that each repetitive invocation of the SPH from the SPH-Z and SPH-V includes steps 4 and 5 as described in Section 1.2. The same is the case for the other repetitive heuristics described in the remainder of this subsection.

A generalization (which appears to be new) of the SPH, SPH-Z, and LPH is to begin with T consisting of the shortest path between any pair of Z-vertices and remaining isolated Z-vertices, and apply the SPH from there. The shortest tree found in this way (when all $p(p - 1)/2$ shortest paths have been tried as initial parts of T) is guaranteed to be at least as good as the tree obtained by the SPH, SPH-Z, and LPH. On the other hand, the worst-case time complexity of this heuristic, in the following referred to as the SPH-ZZ, is $O(p^3n^2)$; all pairs of Z-vertices are to be considered.

A compromise between the SPH and SPH-ZZ, in the following denoted by the SPH-zZ, would be to run the SPH for $p - 1$ initial shortest paths whose one endpoint is a fixed Z-vertex. This reduces the complexity to $O(p^2n^2)$, and yields a solution at least as good as the solution obtained by the SPH.

Since none of the repetitive heuristics is worse than the SPH, the worst-case error ratio for each of them is bounded by 2.

2.3. Kruskal-Based Shortest Path Heuristic (K-SPH). The SPH is very sensitive to the choice of the initial vertex. One possible way to avoid this would be a heuristic related to Kruskal's algorithm for the MST problem [12]. The second step of the K-SPH is as follows:

- *Step 2.* Determine a pair of trees in T closest to one another (ties are broken arbitrarily). Add to T the minimum-cost path joining these two trees.

Such a heuristic was suggested in [17] but to the best of our knowledge it was never implemented nor tested. Furthermore, it remains an open problem whether the worst-case error ratio for the K-SPH is bounded by 2. The close relation of the K-SPH to both SPH and DNH (see the next section) can perhaps make it possible to use worst-case error ratio proofs for these heuristics [20], [21] in connection with the K-SPH.

3. Distance Network Heuristic (DNH). This well-known heuristic was given independently by Kou *et al.* [11], Plesnik [14], and El-Arbi [8]. Its step 2 is:

- *Step 2:* Determine a Z-vertex z' closest to any Z-vertex $\bar{z}$ of a subtree containing an *a priori* fixed Z-vertex z (ties are broken arbitrarily). Add to T the portion of the shortest path between z' and $\bar{z}$ which joins z' and the subtree.

Step 2 corresponds to the iteration of Prim's algorithm for the MST problem [15] of the distance network $D_G(Z)$. In other words, $D_G(Z)$ denotes the complete network with Z as its vertex set, and shortest path lengths in G as costs of edges between pairs of Z-vertices.

The DNH as described above differs slightly from the original version given in [11]. The shortest path added at each iteration in the original version is not required to follow the network constructed so far. Consequently, upon termination, T does not need to be a tree in the original version.

The overall worst-case time complexity of the DNH is $O(pn^2)$ if the computation of shortest paths between all Z-vertices in a network with n vertices is carried out. Several modifications of the DNH which compute T more efficiently (by avoiding shortest paths computation) have been suggested in the literature [25], [22], [13]. However, these modifications do not influence the quality of solutions obtained. Consequently, they are not discussed here.

The DNH has the worst-case error ratio bounded by 2 [11]. More specifically,

$$\frac{c(T)}{c(T_0)} \leq 2\left(1 - \frac{1}{l}\right) \leq 2\left(1 - \frac{1}{p}\right),$$

where l is the number of leaf-vertices in T (bounded by p, the number of Z-vertices).

4. Average Distance Heuristic (ADH). The last well-known PDH, in the following referred to as the ADH, was developed by Rayward-Smith and Clare [16], [17]. Step 2 of the ADH is as follows:

- *Step 2:* Select a vertex v^* which in some way (defined below) can be regarded as most central for the subtrees in T. Add to T the edges on the shortest paths from v^* to the closest and second-closest tree in T.

The centrality measure suggested in [17] is as follows. For each vertex v, sort the subtrees of T in nondecreasing order of their distance to v. Note that if v already belongs to some subtree, this subtree will be the first in the ordered sequence. Suppose that the ordering is $T_1^v, T_2^v, \ldots, T_k^v$. Let

$$f(v) = \min_{2 \le r \le k} \left\{ \sum_{i=1}^{r} \frac{d(v, T_i^v)}{r-1} \right\}.$$

Now v^* is selected among all vertices for which the f-value is smallest. Fortunately, not all $k-1$ terms need to be considered when determining $f(v)$. If $v \in Z$, then $f(v) = d(v, T_2^v)$. If $v \in S$, then the smallest r, $2 \le r \le k$, such that the rth expression is smaller than the $(r+1)$st expression is the value of $f(v)$.

It is evident that the ADH is a generalization of the K-SPH. In the K-SPH the selection of the closest pair of trees in T corresponds to the selection of v^* which minimizes f over all choices of $v \in Z$. Since the K-SPH is closely related to the SPH, it should not be surprising if the computational experience reveals that the ADH is in general superior to the SPH. This is indeed the case.

The worst-case time complexity of the ADH is $O(n^3)$. Furthermore, as recently proved by Waxman and Imase [21], the worst-case error ratio is bounded by the same constant as for the DNH.

5. Reductions. An important step toward finding exact solutions to the instances of the SMT problem is to reduce the original network by identifying

- edges and S-vertices which must belong to the SMT,
- edges and S-vertices which do not belong to the SMT.

When an edge e is known to belong to the SMT, the network can be contracted along e. If parallel edges emerge as a result of such a contraction, only the shortest one needs to be retained. When an S-vertex is known to belong to the SMT, it can be regarded as a Z-vertex. When an edge or an S-vertex is known to be outside of the SMT, it can simply be deleted.

Reductions form a natural approach to accelerate solving the SMT problem. The importance of efficient reductions applied before (and to some extent also during) any of the exact enumeration algorithms for the SMT problem has been thoroughly examined in the literature [2], [4], [7].

On the other hand, such heuristics as the SPH, DNH, and especially ADH can be regarded as applying some of the simple reductions during their iterative greedy march toward a suboptimal solution. As a consequence, use of reductions as a preprocessing phase is of limited importance. As a matter of fact, comparison of performance of the SPH, DNH, and ADH with and without reductions can be indicative of the quality of solutions obtained. Any good heuristic should perform equally well with and without reductions.

For heuristics such as the SPH-Z, SPH-zZ, SPH-V, and SPH-ZZ, which involve $O(p)$, $O(n)$, or $O(p^2)$ repetitive applications of the SPH, such reductions can be of

extreme importance as they reduce the number of vertices. In particular, reductions causing edge-contractions are useful.

For the proof of validity of the reductions below and for additional more elaborate reductions (which seem too complex to be applied in heuristics) the reader is referred to [7].

5.1. Degree Reductions. Any S-vertex s of degree 1 and its incident edge (s, v) can be eliminated. Any S-vertex s of degree 2 and its incident edges (s, v) and (s, w) can be replaced by the edge (v, w) of length equal to the sum of lengths of edges (s, v) and (s, w). If, as a consequence, v and w becomes connected by two parallel edges, only the shorter one needs to be retained. These reductions are of rather limited importance since they only apply to relatively sparse networks.

5.2. Longest-Edge Reductions. An edge (v_i, v_j) can be deleted from the network if the shortest path between v_i and v_j consists of more than one edge. This reduction, in the following referred to as the longest-edge reduction of type I, is not applicable to networks satisfying the triangle inequality. Furthermore, it is not applicable to networks embedded in the Euclidean plane. There is another longest-edge reduction which is applicable to such networks. It is referred in the following as the longest-edge reduction of type II, and is as follows. Consider an edge (v_i, v_j). Suppose that there is a Z-vertex z such that

$$\max\{d(z, v_i), d(z, v_j)\} < c(v_i, v_j).$$

Then the edge (v_i, v_j) cannot be in the SMT and can be eliminated.

5.3. Closest Z-Vertices Reductions. Let z denote a Z-vertex. Let v_i and v_j denote the closest and second-closest neighbors of z. Since G is assumed to be connected, v_i always exists. However, it may happen that v_j does not exist. Then $c(z, v_j) = \infty$. Let z' denote a Z-vertex, other than z, which is closest to v_i. If

$$c(z, v_i) + d(v_i, z') \leq c(z, v_j),$$

then the edge (z, v_i) must belong to the SMT. Consequently, the original problem instance can be reduced by the contraction along this edge. It should be noted that after the contraction, it is not necessary to recompute the distance matrix. The contraction can be carried out if instead of actual distances, their upper bounds are used. Thus, it suffices to redefine $d(z, v_k)$ (and hence $d(v_k, z)$) to $\min\{d(z, v_k), d(v_i, v_k)\}$ for all $v_k \in V$, $v_k \neq z$, v_i, and then delete the row and column corresponding to v_i. It is in particular this test that is responsible for substantial reductions of problem instances. Furthermore, its efficiency increases as the number of Z-vertices grows. This is especially useful since the number of invocations of the SPH from the SPH-Z and SPH-zZ is proportional to the number of Z-vertices. Even more important, the number of invocations of the SPH from the SPH-ZZ is proportional to the square of the number of Z-vertices.

6. Experimental Results. In this section we bring together the various heuristics and compare their computational performance. The only extensive comparison of the SPH, DNH, and ADH was previously done in [17]. Two main issues were considered:

- What is the quality of suboptimal solutions when compared with optimal solutions?
- How do the heuristics perform with respect to one another?

Unfortunately, the results concerning the first aspect were rather uncertain. As remarked in [17], all but one of 19 problem instances tested involved very sparse networks (with $\leq 2n$ edges). Furthermore, no results on how the heuristics perform for various types of networks (see below) were given. Neither was any information concerning the impact of various edge cost distributions provided.

The second aspect was investigated in much more detail. However, all experiments were carried out on randomly generated networks with fixed probability p_e that an edge between two vertices exists, and with fixed probability p_s that a vertex is a Z-vertex. Networks with $p_e = 0.125, 0.25, 0.5, 0.75, 1.0$ and $p_s = 0.125, 0.25, 0.5, 0.75, 0.9$ were then tested (but not all combinations). Edge costs were either uniformly distributed in]0, 1], or with a normal distribution with mean 0.5 and standard deviation 0.125.

The main conclusions of [17] were that the SPH and ADH produced optimal solutions in approx. 50% of problem instances (although as mentioned previously very few and very sparse networks were considered). In this paper we do not compare solutions obtained by the heuristics with optimal solutions.

Concerning the relative performance of heuristics reported in [17], the DNH was found inferior to both the SPH and ADH (although in few cases the DNH performed better than both the SPH and ADH). Furthermore, the ADH failed to get the best solutions (not necessarily optimal) in very few cases (under 1% of cases). On the other hand, the SPH failed to get the best solutions in 8–9% of cases.

The superiority of the ADH is of course due to its elaborate strategy of selecting subtrees to be interconnected. Furthermore, the quality of solutions of the extremely simple SPH depends on the selection of the initial Z-vertex. The question which arises is whether better solutions can be obtained by the SPH-Z, SPH-zZ, SPH-ZZ, SPH-V, LPH, and K-SPH. Will their performance be comparable or better than the performance of the ADH? Are there special types of networks for which some of the heuristics are especially well-suited?

If any of the SPH-Z, SPH-zZ, SPH-ZZ, SPH-V turns out to perform better than the ADH, the next question is whether the reductions can provide computational savings so that run times of these heuristics are comparable.

6.1. Networks. In the following subsections we report on computational results for:

- Random networks with varying number of Z-vertices, varying edge density, and randomly generated edge-costs with varying distribution.
- Euclidean networks with vertices randomly embedded in the plane, varying

number of Z-vertices, varying edge-density, and edge-costs determined by the L_2 distances.
- Grid networks with Z-vertices randomly embedded in the plane, S-vertices determined by intersections of horizontal and vertical lines through Z-vertices, and edges being the horizontal and vertical line segments between the vertices.

All heuristics and reductions were implemented in Pascal and run on a VAX 6410 computer with VMS Operating System 5.2. Unless otherwise stated, reductions are applied to all heuristics.

6.2. Applicability of Reductions. While testing the heuristics, the reductions where applied exhaustively in the following sequence:

- degree reductions followed by the recalculation of distances,
- longest-edge reductions of type I,
- degree reductions followed by the recalculation of distances,
- closest Z-vertices reductions followed by the recalculation of the distances,
- longest-edge reductions of type II,
- degree reductions followed by the recalculation of the distances.

It is by no means claimed that this reduction sequence is optimal. Another sequence and/or additional applications of reductions (and of the closest Z-vertices reductions in particular) would most likely reduce problem instances even further.

Table 1 indicates the impact of reductions on randomly generated networks. It shows, for each $n = 15, 20, 25, \ldots, 75$ and for each ratio $p/n = 0.1, 0.2, \ldots, 0.9$, the following two numbers:

- Percentage of vertices in the reduced network (in relation to the original number of vertices n).
- Ratio between the reduced number p' of Z-vertices and the reduced number n' of vertices in total.

Each entry is an average taken over 21 problem instances with a given n and p/n but with varying edge density (0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0) and varying distribution of edge costs (uniform between 1 and 10, 1 and 100, 1 and 1000, respectively). It can be seen that reductions are quite powerful. The largest reductions in terms of the number of vertices are obtained (as could be expected) for large p/n. New ratios p'/n' seem to stay close to the old ratios p/n (at least for larger n).

Additional tests, not reported in detail here, revealed that the strength of reductions seems to increase as the variation of edge costs increases. No significant variation of the strength of reductions was observed when density changed (except for very sparse networks).

Table 2 shows the same information for Euclidean networks. Networks with $n = 15, 20, 25, \ldots, 60$ vertices were tested. Each entry is based on 21 problem instances, three for each edge density (0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0). It can be seen that reductions in Euclidean networks become more powerful as the number of vertices grows. In general, however, reductions are less powerful than for their

Table 1. Reductions on random networks.

R	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
15	48—0.46	41—0.51	36—0.67	26—0.72	20—0.86	20—0.88	13—0.93	8—1.0	6—1.0
20	60—0.25	42—0.45	50—0.46	37—0.60	23—0.79	21—0.81	19—0.96	12—0.96	5—0.98
25	63—0.24	48—0.38	48—0.40	40—0.56	28—0.73	22—0.73	17—0.78	10—0.97	4—1.0
30	74—0.12	59—0.29	58—0.31	39—0.48	33—0.58	18—0.79	18—0.81	8—0.94	6—0.96
35	71—0.12	63—0.22	53—0.37	41—0.51	30—0.60	25—0.64	14—0.82	8—0.91	4—0.97
40	76—0.11	67—0.21	57—0.33	45—0.44	33—0.51	24—0.59	14—0.76	8—0.94	5—0.97
45	80—0.11	67—0.21	51—0.33	44—0.38	30—0.53	23—0.60	14—0.70	9—0.92	5—0.91
50	81—0.10	72—0.19	61—0.28	50—0.37	33—0.49	25—0.56	14—0.76	9—0.88	4—0.96
55	80—0.11	72—0.18	61—0.28	48—0.37	33—0.46	27—0.52	11—0.80	10—0.84	4—0.98
60	86—0.10	71—0.20	63—0.27	53—0.34	30—0.49	25—0.53	15—0.67	8—0.90	4—0.95
65	84—0.10	76—0.18	59—0.27	54—0.32	32—0.44	25—0.59	14—0.71	8—0.87	3—0.97
70	84—0.09	73—0.19	62—0.26	47—0.33	37—0.43	27—0.49	16—0.68	7—0.86	5—0.93
75	85—0.10	74—0.18	63—0.27	51—0.33	36—0.44	27—0.51	15—0.70	11—0.78	3—0.94

Table 2. Reductions on Euclidean networks.

E	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
15	73—0.28	57—0.43	53—0.51	42—0.62	38—0.73	33—0.79	23—0.86	18—0.94	9—0.97
20	79—0.22	76—0.30	67—0.39	60—0.46	47—0.58	37—0.72	29—0.80	16—0.89	10—0.96
25	82—0.19	78—0.27	67—0.38	61—0.46	50—0.57	34—0.66	30—0.74	21—0.81	10—0.92
30	86—0.12	84—0.20	78—0.29	65—0.39	52—0.52	42—0.62	32—0.70	23—0.80	13—0.90
35	88—0.12	85—0.20	76—0.29	66—0.36	51—0.49	46—0.55	32—0.67	22—0.75	9—0.95
40	92—0.09	88—0.18	79—0.27	68—0.39	60—0.43	50—0.53	35—0.67	26—0.71	11—0.89
45	95—0.10	87—0.17	79—0.25	72—0.39	58—0.43	50—0.53	33—0.67	23—0.71	9—0.89
50	95—0.09	89—0.19	82—0.26	70—0.34	62—0.42	48—0.50	35—0.65	23—0.78	10—0.87
55	94—0.10	89—0.17	81—0.26	70—0.34	57—0.42	45—0.50	35—0.65	23—0.78	11—0.86
60	95—0.08	90—0.17	78—0.24	70—0.33	61—0.41	50—0.50	36—0.62	24—0.71	10—0.86

randomly generated counterparts. This seems primarily to be due to the fact that shortest paths between adjacent vertices consist of the interconnecting edge.

When the vertices are embedded in the plane as in the Euclidean networks but edge-costs are determined by the L_1 distance metric (rather than L_2), similar reductions were observed.

For grid networks, reductions turned out to be of very limited importance and practically had no improving effect. However, specialized reductions for grid networks would most likely change this situation considerably.

6.3. Solutions in Random Networks. Table 3 shows how often one of the heuristics outperformed each of the remaining heuristics. More specifically, the entry in row i and column j indicates how many times the ith heuristic outperformed the jth heuristic. Reductions were applied to all heuristics.

Data in Table 3 accumulate 2457 runs: one for each combination of:

- $n = 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75.$
- $p = \lfloor in/10 \rfloor$, $i = 1, 2, \ldots, 9.$
- $m = \lfloor dn(n-1)/200 \rfloor$, $d = 10, 20, 30, 40, 50, 75, 100.$
- $c = [1, \max]$, $\max = 10, 100, 1000.$

Figure 2 shows how often each heuristic obtained solutions not dominated by solutions of any other heuristic. It can be seen from Table 3 and Figure 2 that the SPH-ZZ and SPH-V perform best. Furthermore, the SPH-V seems to perform slightly better than the SPH-ZZ for smaller networks. For larger networks the SPH-ZZ performs better than SPH-V. This is probably due to the fact that the SPH-ZZ is better able to identify two or more interconnected S-vertices. This phenomenon is more likely to occur in larger networks. The SPH-zZ comes in third, followed by the SPH-Z and ADH. Although the SPH-zZ sometimes outperforms the SPH-V, and the ADH sometimes outperforms the SPH-ZZ and SPH-V, the opposite situation occurs more often. For instance, the ADH outperforms the SPH-ZZ 24 times, but the SPH-ZZ outperforms the ADH 225 times out of 2457.

The frequency of obtaining the best solutions seems to be stable for the SPH-ZZ and SPH-V as n grows. This is neither the case for the ADH nor any other heuristic.

Table 3. Random networks: all-to-all comparisons.

R	SPH	DNH	ADH	LPH	K-SPH	SPH-Z	SPH-zZ	SPH-ZZ	SPH-V
SPH	—	224	29	612	59	0	0	0	0
DNH	28	—	22	563	46	1	9	0	0
ADH	298	445	—	722	273	114	93	24	17
LPH	133	222	56	—	138	51	30	0	4
K-SPH	76	254	17	615	—	8	15	1	3
SPH-Z	265	416	111	705	265	—	63	0	0
SPH-zZ	325	461	158	738	327	124	—	0	13
SPH-ZZ	439	553	225	789	436	213	151	—	30
SPH-V	447	566	218	796	444	209	157	23	—

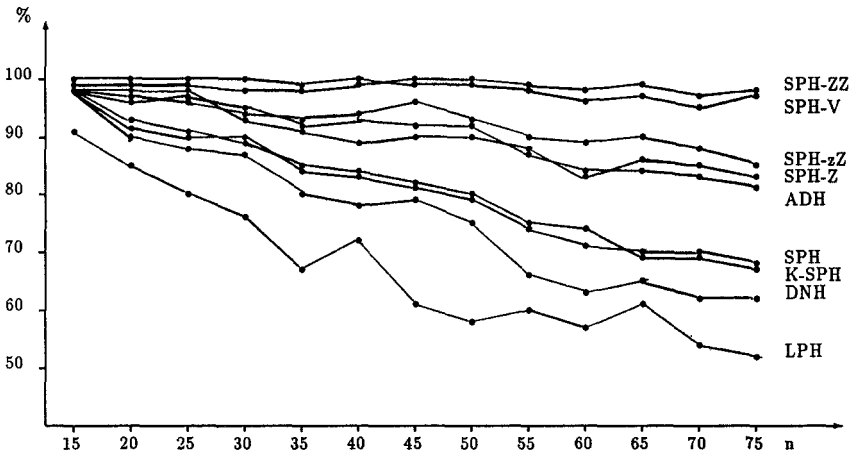

Fig. 2. Random networks: best solution frequency.

No deviation from the ranking of heuristics was observed when best solution frequencies were examined separately for various ratios p/n of the number of Z-vertices over the number of all vertices. As the ratio approaches 1, the differences between solutions obtained by heuristics vanish. For $p/n = 0.9$, all heuristics obtained not less than 90% of best solutions. The differences between solutions are most visible for small p/n.

The frequency of obtaining best solutions seems to be independent of the density of randomly generated networks (for all heuristics).

The performance of the SPH-V, SPH-ZZ, SPH-zZ, and SPH-Z seems to be independent of the variations of edge lengths. The ADH seems to perform best when variations of edge lengths are small.

The relative difference $(c_A - c_V)/c_A$ between the length c_V of the SPH-V solutions and the length c_A of the ADH solutions (when the SPH-V performed better) was on average 1.91%. The maximal relative difference observed was 11.1%. When the ADH performed better, the relative difference $(c_V - c_A)/c_V$ was on average 1.86%. Maximal relative difference was 7.69%.

6.4. Solutions in Euclidean Networks. Table 4 shows how often one of the heuristics outperformed each of the remaining heuristics for problem instances in Euclidean networks. These data accumulate 1890 runs: three for each combination of:

- $n = 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$.
- $p = \lfloor in/5 \rfloor$, $i = 1, 2, \ldots, 9$.
- $m = \lfloor dn(n-1)/200 \rfloor$, $d = 10, 20, 30, 40, 50, 75, 100$.

Figure 3 shows how often each heuristic obtained a solution not dominated by a solution of any other heuristic. It can be seen from Table 4 and Figure 3 that the SPH-V is clearly the best. The SPH-ZZ comes second, the SPH-zZ and ADH third

Table 4. Euclidean networks: all-to-all comparisons.

E	SPH	DNH	ADH	LPH	K-SPH	SPH-Z	SPH-zZ	SPH-ZZ	SPH-V
SPH	—	125	29	800	41	0	0	0	0
DNH	22	—	21	779	34	1	8	1	1
ADH	366	430	—	918	352	283	212	81	38
LPH	165	210	102	—	162	125	42	0	4
K-SPH	49	141	19	805	—	3	16	1	0
SPH-Z	151	235	71	828	146	—	49	0	0
SPH-zZ	375	451	218	906	382	281	—	0	11
SPH-ZZ	585	634	325	986	584	474	277	—	34
SPH-V	649	697	360	1012	649	540	348	106	—

Furthermore, the frequency of obtaining best solutions by the SPH-V and SPH-ZZ seems to decrease slowly as n grows. This is not the case for any other heuristic. Another remarkable observation is that the SPH-zZ is clearly better than the SPH-Z (although they both invoke the SPH the same number of times). In fact, the SPH-Z performs worse than the ADH. This was not the case for random networks.

The differences between the performances of heuristics are more transparent than for random networks. This is perhaps due to the fact that reduction tests were not as powerful.

The relative difference $(c_A - c_V)/c_A$ between the length c_V of the SPH-V solutions and the length c_A of the ADH solutions (when the SPH-V performed better) was on average 1.35%. The maximal relative difference observed was 12.6%. When the

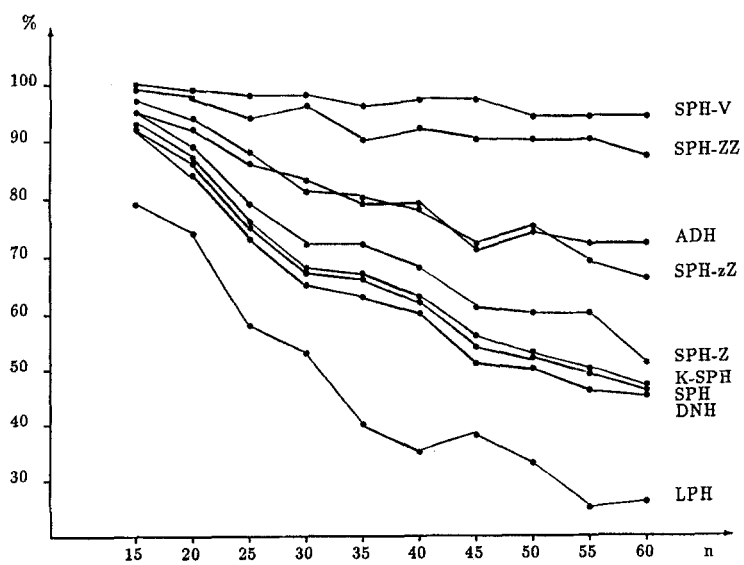

Fig. 3. Euclidean networks: best solution frequency.

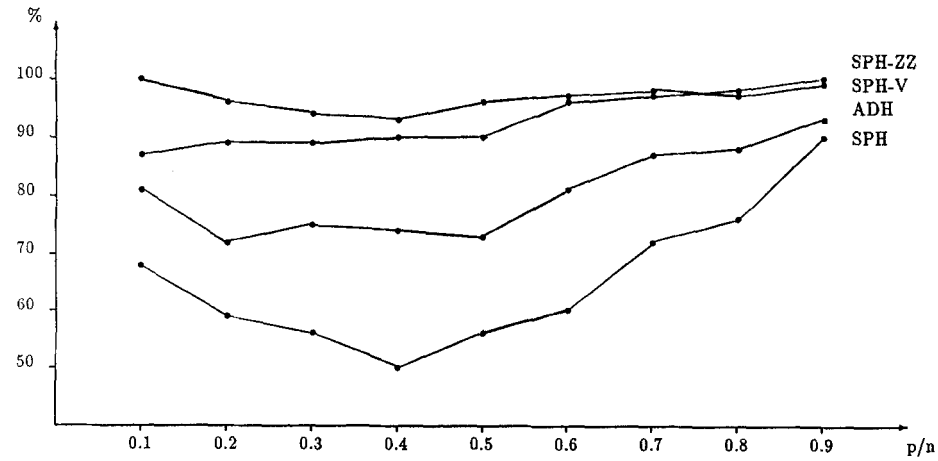

Fig. 4. Euclidean networks: best solution frequency as a function of the ratio p/n.

ADH performed better, the relative difference $(c_V - c_A)/c_V$ was on average 0.81%. The maximal relative difference was 8.09%.

Figure 4 shows the best solution frequency for the SPH-V, SPH-ZZ, ADH, and SPH as a function of the ratio p/n of Z-vertices over all vertices. Figure 5 shows the best solution frequency for the same heuristics but as a function of the edge density. It can be seen that the SPH-V performs better for almost all ratios (with most significant differences for ratios between 0.2 and 0.6) and for almost all edge densities (with most significant difference from the SPH-ZZ for dense networks).

Analogous test runs were carried out for random Euclidean networks with edge lengths equal to L_1 distances. The results essentially follow the same pattern as for L_2 distances.

6.5. Solutions in Grid Networks. Table 5 shows how often one of the heuristics outperformed each of the remaining heuristics in grid networks. These data accumulate 160 runs: 20 for $p = 3, 4, \ldots, 10$ and $n = p^2$.

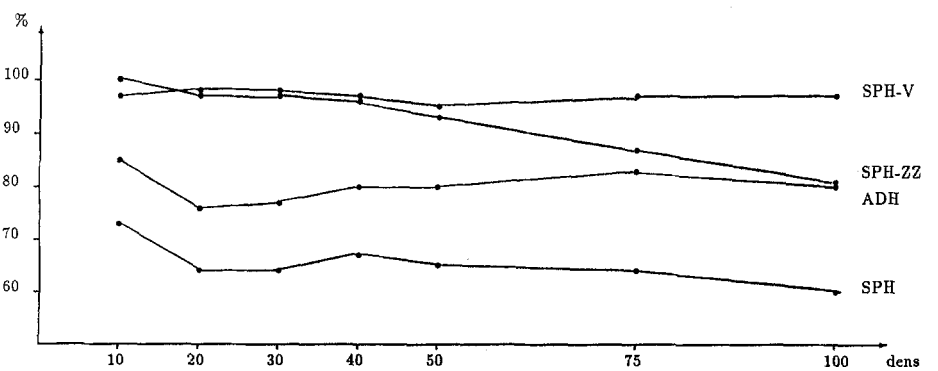

Fig. 5. Euclidean networks: best solution frequency as a function of the edge density.

Table 5. Grid networks: all-to-all comparisons.

G	SPH	DNH	ADH	LPH	K-SPH	SPH-Z	SPH-zZ	SPH-ZZ	SPH-V
SPH	—	45	14	88	7	0	0	0	0
DNH	3	—	12	84	3	0	1	0	0
ADH	99	107	—	109	77	39	64	8	11
LPH	33	38	10	—	25	16	12	0	0
K-SPH	54	80	24	95	—	2	22	0	0
SPH-Z	86	102	46	105	59	—	50	0	0
SPH-zZ	69	90	34	98	46	13	—	0	0
SPH-ZZ	108	115	66	116	91	57	87	—	13
SPH-V	108	115	64	114	89	55	86	2	—

Figure 6 shows how often each heuristic obtained solutions not dominated by solutions of any other heuristic. Both Table 5 and Figure 6 clearly indicate the superiority of the SPH-ZZ over all other heuristics, and the SPH-V in particular. Figure 6 also reveals that the SPH-Z and ADH are next-best. While the frequencies of the best solutions generated by the SPH-ZZ and SPH-V decrease relatively slow as p grows, frequencies of best solutions obtained by the remaining heuristics are either very low or decrease quickly as p grows.

The relative difference $(c_A - c_V)/c_A$ between the length c_V of the SPH-V solutions

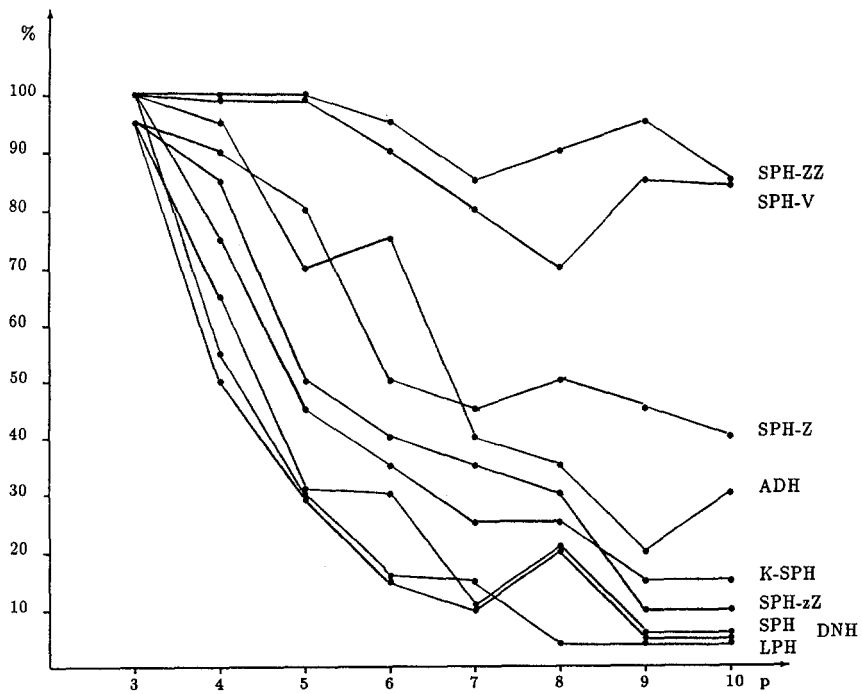

Fig. 6. Grid networks: best solution frequency.

and the length c_A of the ADH solutions (when the SPH-V performed better) was on average 2.37%. The maximal relative difference observed was 6.93%. When the ADH performed better, the relative difference $(c_V - c_A)/c_V$ was on average 1.69%. The maximal relative difference was 3.65%. Note that in only very few cases the performance of the ADH was better than the performance of the SPH-ZZ.

As already mentioned, reductions are of very little importance for grid networks. On the other hand grid networks have relatively few Z-vertices. This in particular favors the SPH-ZZ.

The excellent performance of the SPH-ZZ suggests its particular suitability for problem instances with planar networks embedded in the plane. Indeed, experiments with such networks support this suggestion.

6.6. Computational Times. Table 6 shows computational times (in milliseconds) for the SPH without reductions for $n = 15, 20, \ldots, 45$ and $p/n = 0.1, 0.2, \ldots, 0.9$. Each entry is average over seven test runs with randomly generated networks of varying density (0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0) and with edge costs between 1 and 100. It should perhaps be emphasized that computational times for the SPH and ADH seem to be unaffected by density and edge-cost variations. Furthermore, computational times seem to be independent of the structure of the network (random, Euclidean, grid).

The computational times for the SPH can be used to estimate computational times for any of the repetitive heuristics. Suppose that we want to solve a problem in the Euclidean plane with $n = 50$ vertices and with $p/n = 0.5$. Reductions will (according to Table 2) reduce this problem to one with $n' = 31$ vertices and $p'/n' = 0.42$. Solving a problem of this size by the SPH requires on average 0.5 second + reduction time. Since $p' = 14$, we will need to solve:

- $p' - 1 = 13$ SPH problems when using the SPH-Z and SPH-zZ. Hence the estimated time will be 6.5 seconds + reduction time.
- $n' = 31$ SPH problems when using the SPH-V. Hence the estimated time will be 15.5 seconds + reduction time.
- $p'(p' - 1)/2 = 78$ SPH problems when using the SPH-ZZ. Hence the estimated time will be 39 seconds + reduction time.

Table 6. Computational times (in millisconds) for the SPH.

SPH	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
15	55	64	70	72	72	82	94	92	102
20	110	114	132	147	165	174	177	195	194
25	197	228	277	268	307	311	340	355	370
30	311	410	424	474	508	527	570	584	625
35	521	634	720	737	790	830	871	942	921
40	762	911	1017	1112	1180	1224	1311	1351	1402
45	1110	1388	1452	1572	1675	1734	1817	1877	1970

Table 7. Computational times (in milliseconds) for the ADH.

ADH	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
15	72	100	132	132	147	165	175	172	172
20	127	185	232	271	315	322	317	347	308
25	247	335	451	491	547	555	580	605	588
30	372	600	700	800	910	911	955	948	950
35	667	904	1122	1234	1322	1381	1415	1461	1368
40	918	1288	1532	1750	1892	1938	2044	2052	2032
45	1364	1845	2150	2415	2587	2671	2782	2794	2727

It should be noted that although the worst-case time complexity of the SPH-ZZ is one order of magnitude larger than that of the SPH-V, on average the difference is not as significant. For networks with few Z-vertices, reductions are not powerful but n and $p(p-1)/2$ are comparable. For networks with many Z-vertices, reductions are very powerful whereby computational times for the SPH drop dramatically. Thus, the example given above is essentially worst possible in terms of differences in computational times. If $n = 50$ and $p/n = 0.5$ as before but the network is randomly generated (rather than Euclidean), we have $n' = 17$ and $p'/n' = 0.49$. Solving a problem instance of this size by the SPH will require approx. 0.1 second + reduction time. The SPH-Z and SPH-zZ will require approx. 0.8 second + reduction time, the SPH-V will require approx. 1.6 seconds + reduction time, while the SPH-ZZ will require 3.6 seconds + reduction time. We leave it to the reader to estimate computational times of repetitive heuristics for other values of n and p/n.

Since the ADH is the best of the single-pass heuristics, its computational times (based on the same test runs as for the SPH) are shown in Table 7.

7. Summary and Conclusions. An integrative overview of the path-distance heuristics for the SMT problem on networks has been presented. Within this class, three well-known heuristics (SPH, DNH, and ADH) were examined.

It was clearly shown that two of the heuristics based on repetitive applications of the SPH, called the SPH-V and SPH-ZZ, outperformed all other heuristics on random, Euclidean, and grid networks. Although the worst-case time complexity of these heuristics is $O(pn^3)$ and $O(p^3n^2)$, respectively, these bounds are somewhat misleading. In particular, simple tests proved to be extremely powerful in reducing the size of problem instances before repetitive applications of the SPH in the SPH-V and (in particular) in the SPH-ZZ.

Our extensive analysis on various kinds of randomly generated networks confirmed the superiority of the ADH and the inferiority of the DNH among the single-pass heuristics. Two new simple heuristics (LPH and K-SPH) were shown to be inferior to the ADH. However, the K-SPH seems to be competitive with the SPH and better than the DNH.

Additional heuristic refinements are possible and future testing will further demonstrate new insights into this important area of network design. There are

other possible heuristics within the PDH class and we hope that this paper will provide a constructive framework for their development.

Acknowledgment. We would like to thank the referees for their helpful comments and constructive suggestions.

References

[1] Y. P. Aneja, An integer linear programming approach to the Steiner problem in graphs, *Networks* **10** (1980), 167–178.
[2] A. Balakrishnan and N. R. Patel, Problem reduction methods and a tree generation algorithm for the Steiner network problem, *Networks* **17** (1987), 65–85.
[3] J. E. Beasley, An algorithm for the Steiner problem in graphs, *Networks* **14** (1984), 147–159.
[4] J. E. Beasley, An SST-based algorithm for the Steiner problem on graphs, *Networks* **19** (1989), 1–16.
[5] N. P. Chen, New algorithm for Steiner tree on graphs, *IEEE Symp. on Circuits and Systems*, 1983, pp. 1217–1219.
[6] S. E. Dreyfus and R. A. Wagner, The Steiner problem in graphs, *Networks* **1** (1971), 195–207.
[7] C. W. Duin and A. Volgenant, Reduction tests for the Steiner problem in graphs, *Networks* **19** (1989), 549–567.
[8] C. El-Arbi, Une heuristique pour le probleme de l'arbre de Steiner, *RAIRO Rech. Opér.* **12** (1978), 202–212.
[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
[10] S. L. Hakimi, Steiner problem in graphs and its implications, *Networks* **1** (1971), 113–133.
[11] L. Kou, G. Markowsky, and L. Berman, A fast algorithm for Steiner trees, *Acta Inform.* **15** (1981), 141–145.
[12] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. Amer. Math. Soc.* **7** (1956), 48–50.
[13] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs, *Inform. Process. Lett.* **27** (1988), 125–128.
[14] J. Plesnik, A bound for the Steiner problem in graphs, *Math. Solvaca* **31** (1981), 155–163.
[15] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Tech. J.* **36** (1957), 1389–1401.
[16] V. J. Rayward-Smith, The computation of nearly minimal Steiner trees in graphs, *Internat. J. Math. Ed. Sci. Tech.* **14** (1983), 15–23.
[17] V. J. Rayward-Smith and A. Clare, On finding Steiner vertices, *Networks* **16** (1986), 283–294.
[18] 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, Berlin, 1985, pp. 91–120.
[19] M. L. Shore, L. R. Foulds, and P. B. Gibbons, An algorithm for the Steiner problem in graphs, *Networks* **12** (1982), 323–333.
[20] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Japon.* **24** (1980), 573–577.
[21] B. M. Waxman and M. Imase, Worst-case performance of Rayward-Smith's Steiner tree heuristics, *Inform. Process. Lett.* **29** (1988), 283–287.
[22] P. Widmayer, On approximation algorithms for Steiner's problem in graphs, in G. Tinhofer and G. Schmidt (eds.), *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, Vol. 246, Springer-Verlag, Berlin, 1986, pp. 17–28.
[23] P. Winter, Steiner problem in networks: a survey, *Networks* **17** (1987), 129–167.
[24] R. T. Wong, A dual ascent approach for the Steiner tree problem on a directed graph, *Math. Programming* **28** (1984), 271–287.
[25] Y. F. Wu, P. Widmayer, and C. K. Wong, A faster approximation algorithm for the Steiner problem in graphs, *Acta Inform.* **23** (1986), 223–229.