THE EUCLIDEAN STEINER TREE PROBLEM, IMPLEMENTATION OF A HEURISTIC

T. Hürlimann

Working Paper

August 1994

INSTITUT D'INFORMATIQUE, UNIVERSITE DE FRIBOURG

INSTITUT FÜR INFORMATIK DER UNIVERSITÄT FREIBURG

Institute of Informatics, University of Fribourg

site Regina Mundi, rue de Faucigny 2

CH–1700 Fribourg / Switzerland

Bitnet: HURLIMANN@CFRUNI51

phone: (41) 37 21 95 60 fax: 37 21 96 70

The Euclidean Steiner Tree Problem, Implementation of a Heuristic

Tony Hürlimann, Dr. rer. pol.

Key-words: shortest network, Steiner tree, Eulerian numbers

Abstract: This paper presents a connection between the Eulerian numbers and the numbers of full topologies in the Euclidean Steiner tree problem. This result suggests enumeration strategies, already given in Smith [1992], and attractive neighborhood structures for general heuristics to find the minimal Steiner tree. It also shows how the Euclidean Steiner tree problem fits neatly into the framework of general local search methods (such as simulated annealing or tabu-search) as well as into the paradigm of backtracking and branch-and-bound. A tabu-search heuristic has been implemented with an attractive neighborhood structure which is presented in this paper. Some computational results are given as well. A short historical overview of the Steiner tree problem and its solution procedures are given at the beginning.

Stichworte: kürzestes Netz, Steiner-Baum, Eulersche Zahlen

Zusammenfassung: Dieser Artikel zeigt auf, welche Beziehungen zwischen den Eulerschen Zahlen und der Zahl der vollen Topologien im Euklidschen Steiner-Baum-Problem existieren. Hieraus ergeben sich neue Enumerationsstrategien und attraktive Nachbarschaftsstrukturen für allgemeinen Heuristiken für das Steiner-Problem. Es wird auch gezeigt, dass das Problem des Steiner–Baums gut in das Paradigma der lokalen Suchmethoden (wie des Simulated-Annealing und der Tabu-Search-Methode) sowie der Branch-und-Bound Methode passt. Eine Tabu-Search Methode zusammen mit einer interessanten Nachbarstruktur wurde implementiert und wird hier vorgestellt. Historische Bemerkungen zum Problem und seinen

Lösungsmethoden werden kurz angeschnitten.

INTRODUCTION

> "To conceive a plan and to carry it through are two different things."
> Polya G., How to solve it.

The Steiner Tree Problem is, informally speaking, the problem to find the shortest network that connects a set of points. The problem has an interesting history and a special case of the problem is one of the oldest optimization problems, known also as *the Fermat problem*. Fermat proposed the following problem: find a point in the Euclidean space such that the sum of the distance to three given points is minimal. Torricelli and Cavalieri were the first to give a geometrical construction.

A generalization of the problem was proposed by Simpson, Steiner, Jarník/Klössler, and is reviewed in Kuhn [1975]: find the point such that the sum of the distance to n given points in the plane (or the d-space) is minimal. This problem is also known as the general Fermat problem [Hwang/ Richards/Winter 1992], the 1-Steiner tree problem [Georgakopoulos/ Papadimitriou 1987], or the Star problem [Gallawa 1978]. Georgakopoulos al. gives a $O(n^2)$ algorithm for the plane using new techniques of plane partition based on the Voronoi diagram construction. Since the problem is to minimize the expression $\sum_{i=1}^{n} \sqrt{(x_i - x)^2 + (y_i - y)^2}$ where (x,y) is the unknown point, one can approximate the solution by a simple local search algorithm (the function is convex and has therefore only one minimum).

The 1-Steiner tree problem is not really a – what we might call – interesting generalization of Fermat's problem. The following generalization is a much more difficult and exciting challenge: Given n points in the plane, find the shortest network that connects all of them possibly by introducing a finite number of additional intermediate points. The n given points are called the *terminals* and the additional points – whose locations are unknown – are called *Steiner points*. This problem was popularized in the famous and outstanding book of Courant and Robbins [1941] "What is Mathematics". But they did not give any indications of how to solve the problem except for the three–point problem. They baptized the problem *the Steiner tree problem* although Steiner himself did only some marginal work on the generalized Fermat problem, as many other mathematicians have done too. Today, the problem is called *the Euclidean Steiner tree problem* (ESP) to distinguish it from some other closely related problems.

Two other major research areas have emerged in the sixties to explore Steiner tree problems. The first was proposed by Hanan in 1965. He suggested the *rectilinear Steiner problem*. It is defined like the ESP but with a changed metrics: Manhattan distances are used instead of Euclidean distances. The rectilinear Steiner problem has interesting applications in wiring problems of circuits or buildings. This is actually a intense research domain for designing optimal circuits. A few years later, Hakimi, and independently Levin, presented *the Steiner problem in networks*: Given a graph, the problem is to connect a subset of vertices by a minimal network. If the subset contains exactly two vertices, then the problem is identical to the well known *shortest path problem* in graphs; on the other hand, if the subset includes all vertices of the graph, then the problem is the other famous problem to find *the minimum spanning tree*. Hence, the shortest path problem and the minimum spanning tree problem are both special cases of the general Steiner problem in networks. Both problem are easy (polynomial) problems, whereas the Steiner tree problem in networks is NP-complete in general (a proof is given in Hwang/ Richards/Winter [1992, p 100]). This is true also for the rectilinear Steiner problem [Garey/Johnson 1977] and for the discrete Euclidean Steiner problem, that is the ESP in discrete space where all distances are integer [Garey/ Graham/Johnson 1977]. On the other hand, it is still unknown whether the continuous ESP is NP–complete or much harder [Voss 1990, p 153].

One of the most surprising applications of the Steiner problem in networks consists in testing the theory of evolution [Foulds 1986]. The input is a number of species and their (supposed) relationships in the evolution tree, the output is a improved evolution tree augmented eventually with unknown species (see also the last chapter of Hwang/Richards/Winter [1992]).

There is a vast and still growing literature on the different Steiner tree problems. The state of the art is collected in the monographs of Hwang, Richards and Winter [1992] and Voss [1990]. For good overview papers see also Maculan [1987], Winter [1987], and Hwang/Richards [1992].

This paper deals only with the Euclidean Steiner tree problem. It presents an interesting connection between the Eulerian numbers and the numbers of full topologies of Steiner trees. With this connection, a attractive enumeration algorithm can be developed to find the minimal Steiner tree on any convex metrics that fits neatly into the framework of general heuristics (as simulated annealing or tabu-search) and the paradigm of backtracking and branch-and-bound.

The next section summarizes some definitions and known results about the Euclidean Steiner tree problem (ESP). In the following three sections a brief historical account is given on the solution procedures of ESP. Finally, the last three sections (Eulerian numbers, implementations and computational results) give the results of my own investigations.

DEFINITIONS

Consider a network T interconnecting a set N of n given points in the Euclidean plane, called *terminals*. Any point of T not in N is called a *Steiner point*. The problem is to find the shortest network T connecting all points in N possibly by adding a finite number of Steiner points (with unknown location). Therefore, the problem is also called the *Steiner minimal tree* problem (SMT).

We have the following theorems [Hwang/Richards/Winter 1992]:

– The shortest network must be a tree. Proof: If it were not a tree, one could remove an edge in a cycle, and the new network would also connect all n points.

– The Steiner minimal tree does not contain Steiner points of degree less than three. Proof: Suppose it contains a Steiner point of degree one, then it could be removed together with its adjacent edge without disconnecting the tree and without increasing the length of the network. Suppose it contains a Steiner point of degree two, then one could replace it together with its two adjacent edges by one single edge without increasing the length.

– Any two edges e_1 and e_2 meet at a point p with an angle not less than 120°. Proof: Suppose there is an angle less than 120°, then one can shorten the tree by selecting two points p_1 (on the edge e_1) and p_2 (on the edge e_2), introduce a new Steiner point s, and replace the edges $[p_1,p]$ and $[p,p_2]$ by the three edges $[p_1,s]$, $[s,p_2]$, and $[s,p]$ (see the Torricelli construction below). It also follows from this property that no point in the tree has degree of more than three.

The consequences of these properties are:

– A Steiner point is of exactly degree three. There are at most n–2 Steiner points. Proof: Suppose that the Steiner tree has k Steiner points. Then it has $n+k$–1 edges. But since each Steiner point has three edges and each terminals at least one, the number of edges must be at least $(3k+n)/2$ (division by two because each edge is counted twice). It follows that

$$n + k - 1 \geq (3k + n)/2 \qquad (1)$$

or $k \leq n - 2$ (1a)

– No edge does intersect with each other.

A network – called *Steiner tree* – has a *Steiner topology* if it meets exactly the mentioned degree requirements: (*n* points are given, maximal *n*–2 additional Steiner points are allowed, the degree of all terminals is at most three, and the degree of all Steiner points is exactly three). A topology is called a *full Steiner topology* (FT) if it contains exactly *n*–2 Steiner points. In a full topology the degree of all terminals is exactly one. Steiner topologies which are not full are called *degenerated topologies*. All degenerated topologies can be obtained from a full topology by edge contractions between Steiner points or Steiner points and terminals. If FT is a full topology then D(FT) is denoted as the set of all degenerated topologies obtained by edge contraction from FT.

Clearly, the minimal connecting network is a Steiner tree. For each full Steiner topology FT together with all topologies in D(FT), there exists at most one minimal tree (that is with minimal length), called the *relatively minimal tree* (for a proof see [Hwang/Richards/Winter 1992, p 7]). This follows from the fact that – given a full topology FT – the length of the tree is a convex function with the terminals as given points and the Steiner points with unknown locations. Hence, the relatively minimal tree of all Steiner trees in the set D(FT) – for a given FT – can be found by a local search method, since there exists only one minimum to determine the locations of the Steiner points in the minimal tree; in other words, the relatively minimal tree of D(FT) is unique. This is an important fact and leads directly to the result that it is sufficient to enumerate all full topologies only to find the minimal Steiner tree, which is the smallest of all relatively minimal trees. This procedure is essentially the best algorithm we know today to find the minimal Steiner tree for the ESP in general.

Before exposing this procedure, it is instructive to present several other methods that have emerged in the already 40-year history of solving the ESP. This short historical account shows the tremendous progress that has taken place in solving difficult problems.

THE STEINER TREE PROBLEM WITH 3 POINTS IN THE PLANE

The first special case, where *n*=3, was solved already in the 17th century by Torricelli and Cavalieri by means of a geometric construction. Given three points A, B, and C in the plane, find a point S such that the length AS+BS+CS is minimal. From the properties above it follows that there exists at most one Steiner point S and that the three angles in S are all exactly 120°. Therefore, we have a single full topology and three degenerated cases (the three degenerated cases arise when one of the edge AS, BS, or CS has collapsed). Torricelli's construction finds the unique point S (Figure 1) by the following procedure:

1 if any of the three angles of the triangle ABC is greater or equal than 120° then S is the corresponding point (A, B, or C) (the three degenerated cases)

otherwise

2 construct a equilateral triangle ABD on the opposite side of C

3 draw the circle Ci through the points A, B, and D

4 connect D and C by a line DC

5 Let S be the point which intersects the line CD and the circle Ci. The point S is the Steiner point of the problem. (A proof is found in Courant and Robbins [1941]). The length of the Steiner tree is identical to the segment CD. This was found by Simpson, therefore CD is also called a *Simpson line*.

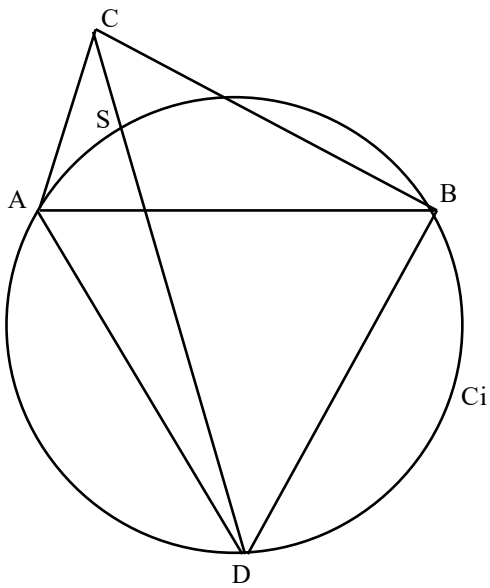


Figure 1 (Torricelli's construction of S)

Clearly, there exist three different Simpson lines, depending on which side

of the triangle ABC the equilateral triangle is constructed. It is easy to show that the three Simpson lines intersect in a single point which again is the Steiner point S.

Figure 2 gives an overview of the Steiner tree problem with 3 terminals. Let A, B, and C be the three terminals. Now, let A and B be some fixed points and move C around in the half-space above the line AB. One can partition the half-space into four subspaces (marked as 1, 2, 3, and 4 in Figure 2). If C is inside the subspace 1, then the Steiner point S is identical to point A, since the angle at A is greater than 120°; if C is in the subspace 3, then the Steiner point S is identical to point C, since the angle at C is greater than 120°; if point C is in the subspace 2, then the Steiner point S is B, since the angle at B is greater than 120°. These three cases cover the three degenerated cases. Otherwise (if C is in the subspace 4) then the Steiner point S is the intersection between the circle and the line CD, as given by Torricelli's construction.

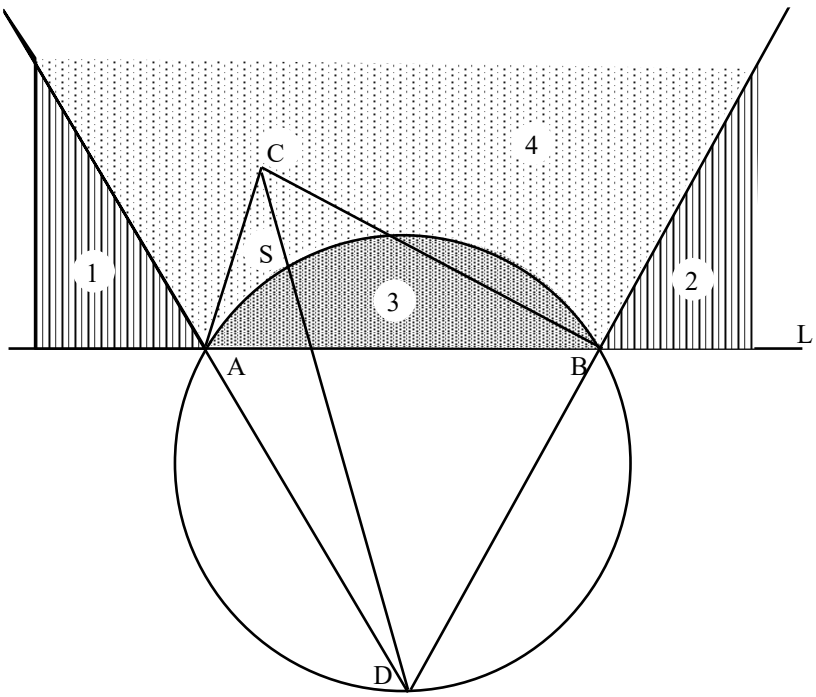


Figure 2 (Gilbert & Pollak, 1968)

Since point S is unique and the function $\sum_{i=1}^{n} \sqrt{(x_i - x)^2 + (y_i - y)^2}$, that has to be minimized, is convex, one can also use a simple local search method.

A third method was proposed by Smith [1992]. All three methods have been implemented (see below).

ANALOG AND MECHANICAL SOLUTIONS OF THE ESP

Several analog or mechanical techniques that use natural systems' tendencies to find minimum energy configurations have been presented to find the minimal Steiner tree or at least a good approximation. Miehle [1958] presents a very nice mechanical model with wheels and strings (Figure 3). He placed fixed wheels on a sheet of wood that represent the terminals. Movable wheels represent the Steiner points. Then he connected all wheels by a single string. When he pulled the two ends of the string, the movable wheels moved to the place in such a manner that the string length became minimal. The main drawback of this model is that the topology is fixed in advance. Hence, this procedure solves only the easy part of the Steiner tree problem; but the difficult part is to find the right topology.

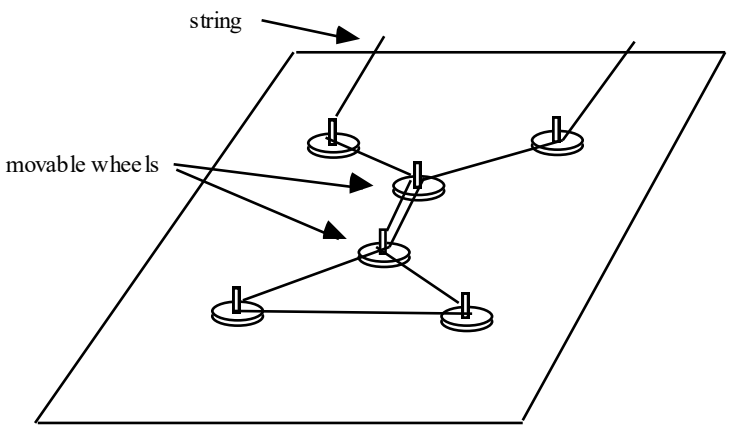


Figure 3: a mechanical model (Miehle)

The soap model – another analog model – is well known and is part and parcel of the folklore (Figure 4). Pegs (representing the terminals) connect two Plexiglas-plates. A fine soap film forms connecting the pegs with a length as minimal as possible. The projection of the film on a plate gives generally a good solution of the Steiner tree problem. The Steiner points are located at the points where the soap film splits. The advantage of this model resides in the possibility to impose additional constraints, such as minimal networks on a hilly surface or in surfaces with holes (lakes). Holes, for example, are neatly bypassed by the soap films.

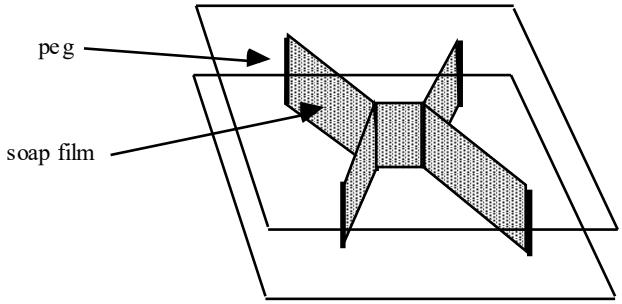


Figure 4: a soap model

Other analog solutions (such as Figure 5) have been presented. But all of them have the disadvantage that they do not necessarily find the optimal solution. This is not so surprising since the problem is very difficult and has many local minima. More surprising is the fact that there *are* analog models that find quite good solutions. The soap model, for example, gives sometimes solutions which are as good as those of the best and fastest algorithms we presently know.

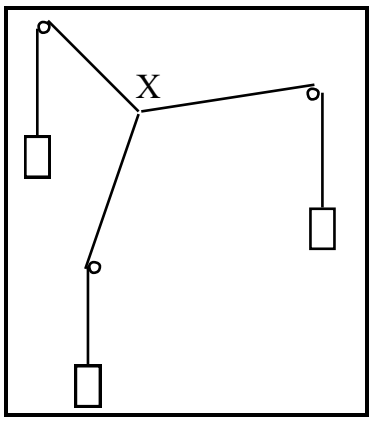


Figure 5: another mechanical model (Polya 1954, Vol. I p148)

As a result, all analog models have the same limitation as any general heuristic that can be implemented on a computer: they get stuck at a local minimum.

In the next section, the most known exact algorithms and heuristics of the ESP are briefly presented.

BRIEF HISTORY OF ALGORITHMS FOR THE ESP

As far as I can see, the solution procedures to find the minimal Steiner tree in the Euclidean metrics can be grouped into four classes: The class based on Melzak's algorithm and its variants, Smith's [1992] branch–and–bound procedure, the heuristics based on the minimal spanning tree, and the heuristics based on some partition of the plane such as the Voronoi diagram. There also exist powerful decomposition theorems to split a problem of *n* terminals in several smaller ones (see [Gilbert/Pollak 1968], [Cockayne 1970], and [Hwang/Song/Ting/Du 1988]). We do not go into this decomposition theorems as we concentrate on undecomposable problems.

Most exact procedures for the Euclidean Steiner tree problem in the plane (2-ESP) rely on the idea of the geometric algorithm of Melzak [1961], which is a extension of Torricelli's construction. The history of all these variants would be an interesting case study for an exemplifying algorithmic progress over the last 30 years. The first computer program based on the idea of Melzak for finding SMT in the plane was written by Cockayne and Schiller [1972]. The procedure consists in applying Melzak's algorithm to every component of a partition of the *n* terminals. Since there are exponentially many partitions and Melzak's algorithm itself is exponential, this implies a super-exponential algorithm to find the minimal Steiner tree, although several decomposition procedures can be applied and most partitions are quickly found to be "infeasible". A comprehensive overview of Melzak's algorithm can be found in [Bern/Graham 1989]. It is remarkable that Melzak's algorithm (which is exponential as I said) can be replaced by a linear algorithm which has been found by Hwang [Hwang 1986] (see also Hwang/Richards/Winter 1992, p 23).

Winter [1985] implemented the GEOSTEINER program which was probably the most efficient program 10 years ago. Cockayne and Hewgill [1986] improved it and implemented EDSTEINER86 and later [1992] EDSTEINER89.

These programs can find the minimal Steiner tree with the following number of terminals.

Cockayne and Schiller [1972]	7
Winter [1985]	15
Cockayne and Hewgill [1986]	17, but up to 30 for most problems
Cockayne and Hewgill [1992]	17, up to 100 for some problems

The last two lines have to be taken with a grain of salt, although

EDSTEINER89 is certainly one of the most efficient program to find minimal Steiner trees in the plane. It was implemented in FORTRAN and consists of 120 pages of source code! Nevertheless, it is important to notice that Torricelli's construction is only feasible in the two-dimensional Euclidean space. Therefore, Melzak's algorithm and its modern linear counterpart by Hwang, both of which are based on Torricelli's construction, break down completely for Steiner tree problems in any other than the two-dimensional Euclidean space. Furthermore, some nice decomposition procedures can be applied for most randomly generated 2-ESP. This is not the case in general: R.L. Graham has shown that there are Steiner tree problems (called ladders) which are not decomposable. So, the state of the art to find the optimal solution still does not go far beyond 15 terminals.

Another exact algorithm was presented by Smith W.D. [1992]. Its is not confined to Steiner tree problems in the plane, it can be applied to any convex metrics. The algorithm contains two steps:

Step 1: Enumerate all full topologies

Step 2: For each topology in Step 1 find the relative minimal tree

Step 1 is related to the theorem that – given a single full topology FT – the relative minimal tree of D(FT) is unique (see also [Trietsch/Hwang 1990]). It follows that we only need to enumerate all full topologies. For step 2 of the algorithm, Smith proposes an iterative numerical method, where at each iteration all Steiner points are updated simultaneously. This can be done by solving a system of $2n-4$ linear equations. Let (x_k,y_k) be the coordinates of the (unknown) Steiner point k; let (x_j,y_j) be the coordinates of the (known) terminal j; at the i=0,1,2,..., step of the iteration, one has to solve the following linear system:

$$x_k^{i+1} = \frac{\sum_j \frac{x_j^{i+1}}{\Delta_{jk}}}{\sum_j \frac{1}{\Delta_{jk}}} \qquad y_k^{i+1} = \frac{\sum_j \frac{y_j^{i+1}}{\Delta_{jk}}}{\sum_j \frac{1}{\Delta_{jk}}} \qquad (2)$$

where each summation is over all j such that $[j,k]$ is an edge in the tree and where Δ_{jk} is the (convex) distance between point j and point k. Smith proved that the iteration converges geometrically to the unique relative minimum tree. Furthermore, the linear system (2) can be solved in linear time since the system can be permuted to a tridiagonal form and a special Gaussian elimination can be applied. The great advantages of this algorithm is that

– it is not confined to the two-dimensional space

– it does not need to keep track of all partitions

– and, consequently, it is easy to implement and to include into the general framework of branch–and–bound methods.

Since Smith's algorithm is more general, it is evident that it is less efficient for the ESP in the plane than the most advanced of the “Melzak family”.

Many heuristics are based on the minimal spanning tree which can be constructed in O(n log n) steps in the plane. The length of the minimal spanning tree produces also a upper bound for the minimal Steiner tree. There is an interesting theorem that also gives a lower bound on the length of the minimal Steiner tree in the plane. The theorem says that the length of the minimal Steiner tree l_S cannot be smaller than 86.6% of the corresponding length of the minimal spanning tree t_M. This ratio is called the *Steiner ratio* $\rho(n)$ and we have therefore the following lower bound

$$\rho(n) = \frac{l_S}{l_M} = \frac{\sqrt{3}}{2} \approx 0.866 \qquad (3)$$

It is easy to see that the Steiner ratio can be attained – just take an equilateral triangle – but to prove that there does not exist a smaller lower bound is much harder. In fact, the bound was conjectured by Gilbert/Pollak in 1968 and has only be proved recently by Du/Hwang [1990]. A proof is also found in Hwang/Richards/Winter [1992].

Chang [1972] was one of the first to propose a O(n^4) heuristic based on the minimal spanning tree:

Step 1: Construct a minimal spanning tree

Step 2: iteratively add locally Steiner points that shorten the tree.

Step 2 can be done in many ways. One of them is presented below to find an initial solution for the local search method. Chang gave experimental results which showed a typical saving of 3% over the minimum spanning tree (p 709).

Another heuristic is based on the Voronoi diagram and its dual, the Delaunay triangulation. [Smith J.M./Lee/Liebman 1981]. Given n points in the plane, the Voronoi diagram is the partition of the plane into n (not necessarily finite) polygons – such that all points inside the i-th polygon are closest to the i-th point. Surprisingly enough, there exists an O(n log n) algorithm to construct the Voronoi diagram in the plane. The Delaunay triangulation – which is a triangulation such that two points are linked by an edge if and only if the

corresponding two polygons in the Voronoi diagram are neighbors – can be obtained in O(n) steps from the Voronoi diagram. The heuristic of Smith/Lee/Liebman [1981] which is based on this construction is very efficient and is probably one of the quickest ways to find a good approximation of the minimal Steiner tree. It first constructs the triangulation, and within each triangle the local optima, if they exist, are found. In a second step these local optima are linked together in a similar way as Kruskal's minimum spanning tree algorithm: the triangles are placed into a priority queue, based on the l_S/l_M reduction ratio. The suboptimal Steiner tree is then constructed by taking these triangles one by one using a disjoint-set-union procedure. The mean reduction over the minimum spanning tree length of randomly generated problems was reported by Smith J.M. al. as 3.13%. The overall complexity of the algorithm is O(n log n).

The experimental results of Chang [1972] and Smith/Lee/Liebman [1981] together with some theoretical results given by Gilbert/Pollak [1986] suggest that an average reduction of the length of the minimal Steiner tree over a minimal spanning tree for a randomly generated set of terminals is about 3%.

EULERIAN NUMBERS AND FULL TOPOLOGIES

It has been shown in the first section that it is sufficient to enumerate all full topologies and to apply a local search method to find the unique minimum for all full topologies in order to find the minimum Steiner tree. The question, therefore, arises of how many full topologies exist. Let $f(n)$ be the number of full topologies with n terminals and, therefore, n–2 Steiner points. Every terminal is connected to a Steiner point and every Steiner point has degree three as shown above. Clearly we have $f(2)=1$ and $f(3)=1$. Let F_n be a full topology with n terminals. To get a full topology F_{n+1} with n+1 terminals, one has only to replace any edge [a,b] (with the endpoints a and b) in F_n with a newly introduced Steiner point s connected to the three points a, b and c, where c is the (n+1)th terminal in F_{n+1} (see Figure 6).

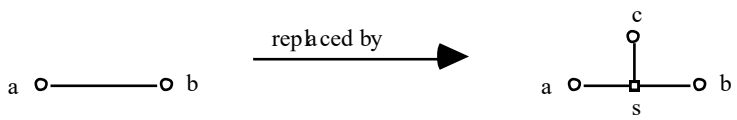


Figure 6

Since there are 2n–3 edges in F_n, we have the recurrence formula

$$f(n+1) = (2n-3)f(n) \qquad (4)$$

which has the solution

$$f(n) = \frac{(2n-4)!}{2^{n-2}(n-2)!} \qquad (5)$$

or

$$f(n) = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-5) \quad (\text{for } n \geq 3) \qquad (6)$$

The following table gives some values for small numbers *n* of terminals.

n	3	4	5	6	7	8	9	10	11
f(*n*)	1	3	15	105	945	10,395	135,135	2,027,025	34,459,425

f(*n*) is a exponentially growing function. It is therefore impracticable to enumerate all full topologies for even the fastest computer unless *n* is very small.

Formula (6) suggests the following enumeration scheme and data structure for full topologies: They can be enumerated (with $n \geq 4$) by a $n-3$ component vector, whose i-th entry a_i ($1 \leq i \leq n-3$) is the integer $0 \leq a_i \leq 2i$ [Smith W.D., 1992, p 143]. As an example, consider all fifteen full topologies of Steiner trees with 5 terminals: they can be represented by the following 15 2-component integer arrays

00, 01, 02, 03, 04, 05, 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 24, and 25.

This data structure has the advantage of very concise storage of a full topology. Furthermore, it is quite easy to construct a graphical representation of a full topology – it can be done in linear time. Winter [1985] used another data structure that could be used directly as a graphical representation of a full topology, he called his data structure “extended binary tree”. In fact all full topologies can be represented by binary trees extended by a extra super-root and submitted to some constraints. Since we do not use Winter's data structure, the interested reader may find the details in Winter [1985].

We use another graphical representation. Consider the unique full topology with $n=3$ (Figure 7). Terminals are represented by circles and Steiner points by squares.

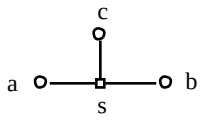


Figure 7

Given the unique full topology with three terminals, we can construct the three full topologies with four terminals as following: replace each edge in Figure 7 (there are three) as indicated by Figure 6. This produces the three topologies in Figure 8.

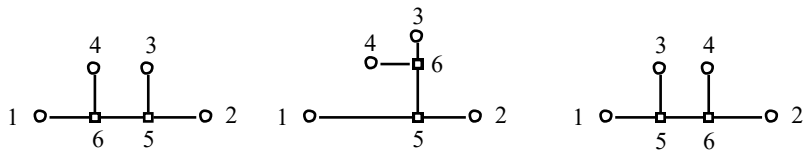


Figure 8

We adopt the convention that a additional terminal has to be on the left and on the upper side of the split edge. Since there are five edges in each of the full topologies in Figure 8, each will generate five full topologies with 5 terminals. Figure 10 shows all 15 topologies of n=5. It is now easy to see how we can construct the graphical representation from the n–3 component vector structure:

– Step 1: construct the tree in Figure 7, (a tree with three terminals and one Steiner point).

– Step 2: for each component i from left to right do the following: insert a Steiner point together with its edge and the (i+4)-th terminal in one of the 2n–3 places following the integer at the i-th component (for the first component that contains the integer 0, 1, or 2, this produces the trees in Figure 8): If the integer is 0, then insert the edge near the terminal 1; if the integer is 2i, then insert the edge between the next two Steiner points; if the integer is 2i–1, then insert the edge on the first vertical edge from the right; and so on.

This procedure produces the tree in Figure 9 from the 7-component vector «0000000» which represents a full topology of ten terminals.

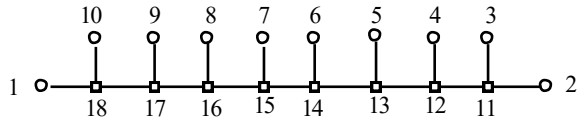


Figure 9

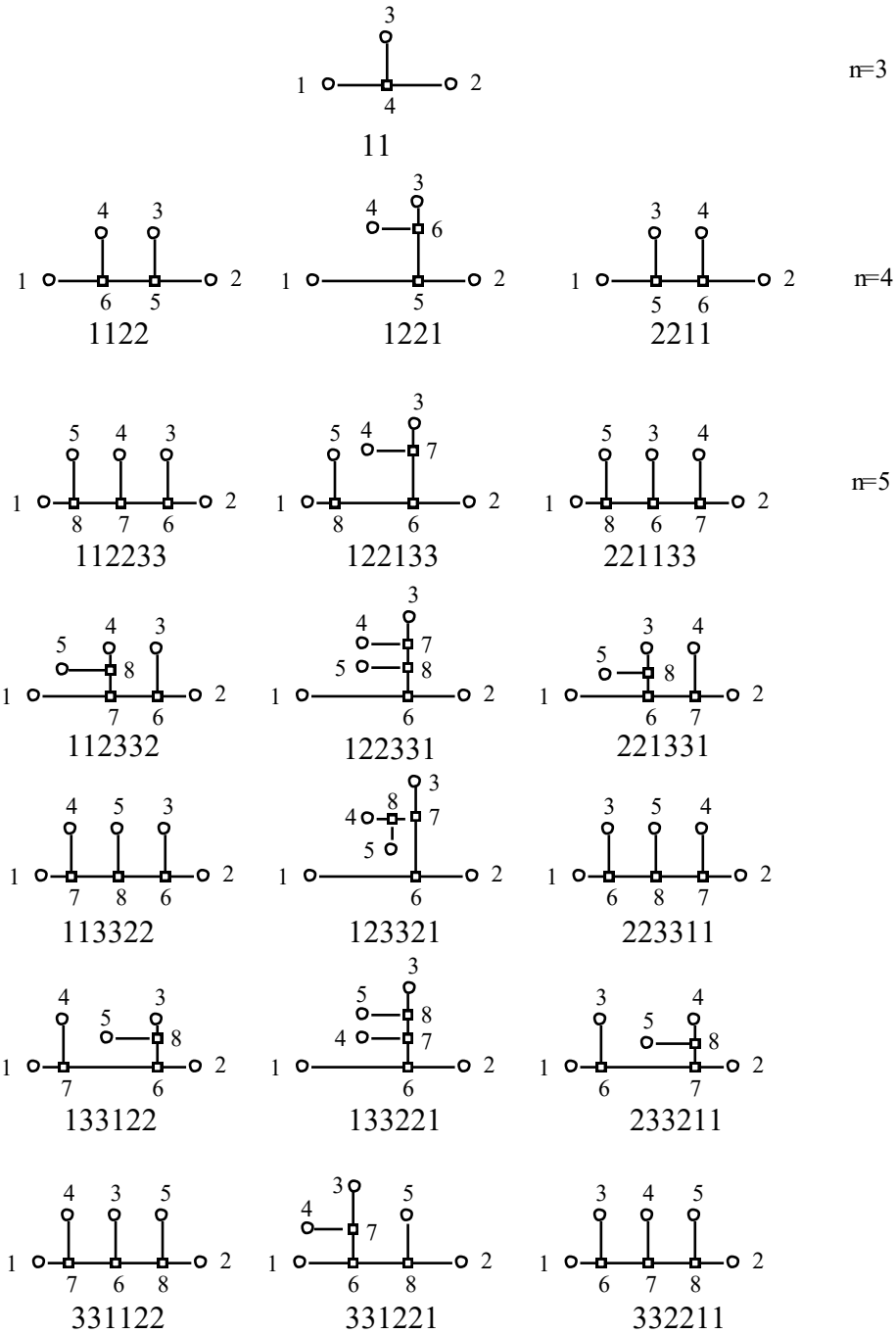


Figure 10 All full topologies for n = 3, 4, and 5

There exists an interesting connection between the Eulerian numbers [Graham/Knuth/Patashnik, p 253ff] and the numbers of full topologies in the Steiner tree problem, that was – as far as I know – not mentioned before in the literature.

But first, let's introduce the Eulerian numbers defined in Graham/Knuth/ Patashnik p 253ff: The *(first-order) Eulerian numbers* are denoted by $\left\langle {n \atop k} \right\rangle$ and produce the Euler's triangle similar to the Binomial numbers which produce the Pascal's triangle. $\left\langle {n \atop k} \right\rangle$ is the number of permutations $\rho=\pi_1\pi_2...\pi_n$ of { 1, 2, ... ,n } that have *k* ascents, namely, *k* places where $\pi_j < \pi_{j+1}$. For example, the eleven permutations 1324, 1423, 2314, 2413, 3412, 1243, 1342, 2341, 2134, 3124, and 4123 have two ascents, hence $\left\langle {4 \atop 2} \right\rangle = 11$; one permutation (4321) has zero ascents, hence $\left\langle {4 \atop 0} \right\rangle = 1$; a single permutation, namely 1234, has three ascents, hence $\left\langle {4 \atop 3} \right\rangle = 1$; the remaining 11 permutations have one ascent. The Eulerian numbers satisfy the following recurrence relation [Graham/ Knuth/ Patashnik 1990 p 254]

$$\left\langle {n \atop k} \right\rangle = (k+1)\left\langle {n-1 \atop k} \right\rangle + (n-k)\left\langle {n-1 \atop k-1} \right\rangle \qquad (7)$$

This recurrence relation is easy to prove: Each (n–1)-permutation $\rho_{n-1} = \pi_1\pi_2...\pi_{n-1}$ leads to *n*-permutations of { 1, 2, ... , n } if we insert the new element *n* in all possible ways. Inserting *n* before position *j* in π yields the *n*-permutation $\rho_n=\pi_1\pi_2...\pi_{j-1}n\pi_j...\pi_{n-1}$. The number of ascents is the same as in ρ_{n-1}, if $\pi_{j-1} < \pi_j$ or $j = 1$; it is one greater, if $\pi_{j-1} > \pi_j$ or $j = n$.

Replacing *n* by $2n–1$ in one place of the recurrence relation (7) leads to the *second-order Eulerian numbers*, denoted by $\left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle$. By definition we have

$$\left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle = (k+1)\left\langle\!\left\langle {n-1 \atop k} \right\rangle\!\right\rangle + (2n-1-k)\left\langle\!\left\langle {n-1 \atop k-1} \right\rangle\!\right\rangle \qquad (8)$$

The second-order Eulerian numbers have a curious combinatorial interpretation: if we form permutations of the multiset { 1, 1, 2, 2, ... , n, n } with the special property that all numbers between the two occurrence of *m* are greater than *m*, for $1 \le m \le n$, then $\left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle$ is the number of such permutations that have *k* ascents. For example, there are six permutations of { 1, 1, 2, 2, 3, 3 } with two ascents, namely 112233, 112332, 133122, 122133, 122331, and 123321, hence $\left\langle\!\left\langle {3 \atop 2} \right\rangle\!\right\rangle = 6$. The multiset { 1, 1, 2, 2, ... , n, n } has a total of

$$\sum_k \left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle = (2n-1)(2n-3)\text{K}\ (1) \qquad (9)$$

suitable permutations, because the two occurrences of n must be adjacent and there are $2n-1$ places to insert them within a permutation for $n-1$.

But (9) is nothing else than the number of full topologies with $n+2$ terminals. The correspondence between the combinatorial interpretation and the full topologies can be seen in Figure 10: below each graph the corresponding permutation of the multiset is given. The connections are as following:

– each number pair (11, 22, ... , nn) corresponds to an edge not on the path from vertex 1 to vertex 2.

– the two numbers identify the end-points of an edge. The graphical structure can directly visualize the permutation.

THE LOCAL SEARCH HEURISTIC

Based on this enumeration scheme, a new promising local search heuristic will be presented in this section. Since the Euclidean Steiner tree problem is a very difficult problem, a successful local search heuristic to find a good approximation of the minimal tree is based on the following components:

1 find one or several “good” starting solutions, i.e. to find full topologies that produce good relatively minimal Steiner trees.

2 determine a efficient neighborhood structure between full topologies for navigate in the solution space of all full topologies effectively.

3 establish a procedure to escape from local minima, such as a particular tabu-search scheme.

4 once a full topology, say FT, is given, find efficiently the relatively minimal Steiner tree in D(FT).

The four components are minimal requirements for a successful implementation: just starting from a randomly generated full topology, for example, was only successful for problems up to 10 terminals (which is quite good); a bad neighborhood structure, on the other hand, did not find much better solutions even if the starting solution is good; neglecting the escape procedure just gives a local minimum; and an inefficient procedure to produce the relatively minimal Steiner tree would costs too much time.

Finding a good starting topology is not so trivial. Of course, the minimal

spanning tree gives – most of the time – a useful indication on how the starting full topology could look like. There are many different possibilities for generating a full topology from the minimal spanning tree solution. The solution which has been adopted for the implementation is as follows:

– Step 1: find a minimal spanning tree (MST) of all terminals.

– Step 2: define a node in MST with degree 1 as the root the MST-tree

– Step 3: traverse the MST-tree recursively in pre-order; at each node sort the leaving edges counterclockwise (such that the corresponding adjacent nodes will be traversed in that sorted order). Store the pre-order in a permutation P.

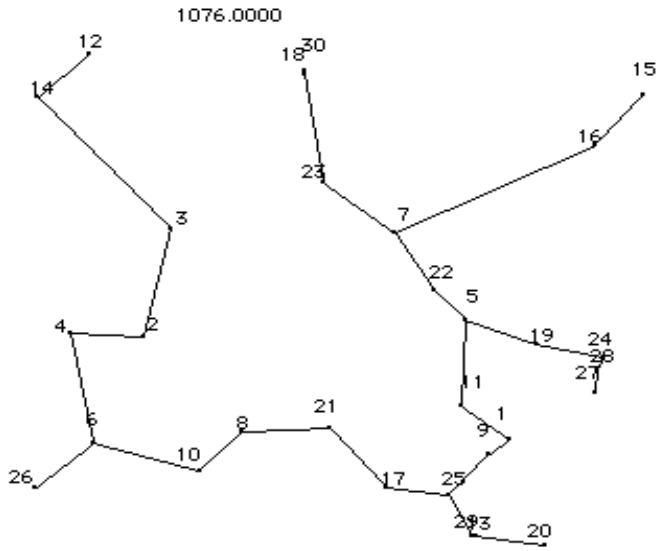


Figure 11

Figure 11 shows a minimal spanning tree of 30 points. The procedure chooses node 12 as root node. At node 6 the branch pointing to 26 is traversed before the branch pointing to 10 (counterclockwise traversal at each node).

– Step 4: permute the nodes in pre-order stored in permutation P (see Figure 12), except root (12) which gets the label 1.

– Step 5: The path from node 1 to node 2 is now denoted the base line in the full topology. If all nodes would be on the base line (this is not the case in Figure 12), then the topology, represented by the n–3 component vector containing n–3 zero entries, would be a good full topology (see Figure 9 for 10 terminals). Since there are branches departing from the base line (from node 26 branch to node 25 in

Figure 12 for example), the component vector has to be adjusted for these nodes. Increasing integers are now attributed recursively to the nodes of the departing branches (see Figure 13). The integers modify the corresponding entry in the n–3 component vector (the details are somewhat technical, and not worthwhile to be described here). The corresponding full topology will produce a first solution for the Steiner tree (see Figure 14).

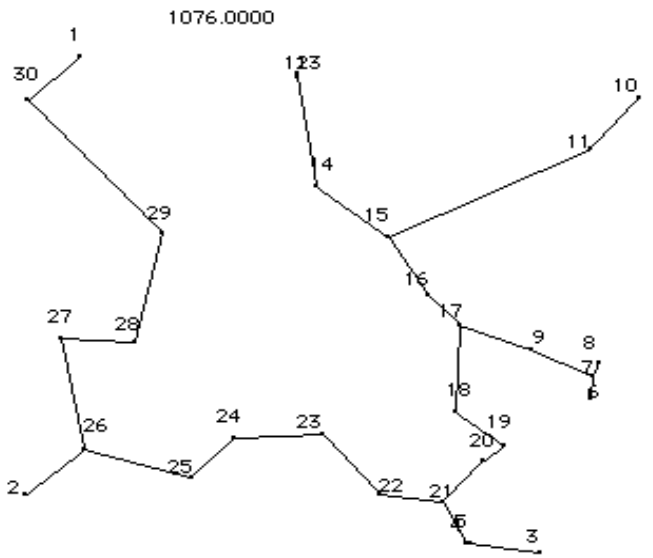


Figure 12 (nodes sorted in pre-order traversal)

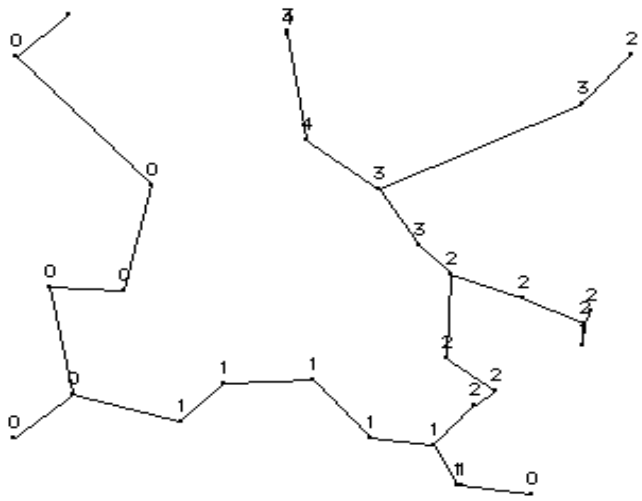


Figure 13

In Figure 14 the 30 terminals are the small circles, the small squares are newly introduced Steiner points. The Steiner tree in Figure 14 is not the

minimal Steiner tree. This can be seen clearly in node 17. But its length is already smaller than the length of the minimal spanning tree.

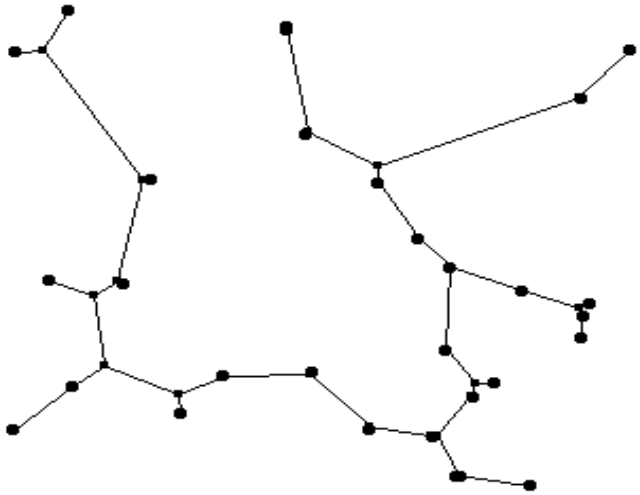


Figure 14 (starting Steiner tree solution for the local search)

Once a starting solution is given, a local search method is used to find better solutions. Thus, a neighborhood structure for full topologies has to be defined. The following neighborhood structure is proposed: At each Steiner point S three branches of the tree T meet. To obtain a neighbor, one has to remove any of the three branches B together with the Steiner point S and attach it at an adjacent edge within the remaining tree T\B. Figure 15 shows the four neighbors when detaching the branch B.

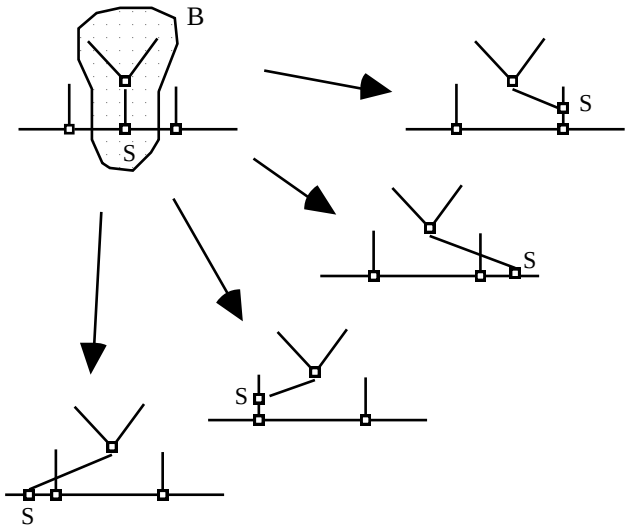


Figure 15

For each Steiner point one can produce maximally 12 neighboring structures. Unfortunately, this neighborhood structure is not so easy to implement if an (n–2)-component vector that represents a full topology is given. Therefore, another somewhat similar structures was adopted: Given a (n–2)-component vector, a neighboring full topology can be obtained by applying the following operation on the *i*-th component c_i:

$$c_i^{new} = (c_i^{old} \pm 2) \bmod (2i)$$

In that way, a neighboring full topology can be easily obtained just by changing one integer in a component. Each full topology has therefore 4(*n*–2) neighboring full topologies. For each full topology FT, we calculate all 4(*n*–2) relatively minimal Steiner trees of N(FT) – if they exist – and retain the tree with the minimal length. Applying this procedure repeatedly will find a locally minimal solution.

The presented neighborhood structure also gives an easy to implement variant to escape local minima: If the best neighbor leads to a minimal tree greater than the actual tree then its solution is accepted as the new actual solution and the backstep to the old solution is blocked just by fixing the corresponding component entry for the next *k* steps in the iteration. This method is essentially the tabu-search method described in Glover [1989 & 1990]. This procedure was quite successful for Steiner problems up to 10 points as one could start from a randomly generated starting solution to get most of the times a supposedly optimal tree. However, the procedure was not at all successful for bigger problems: For those the procedure usually got stuck in the first local minimum found.

The fourth part of the implementation consists in finding the relatively minimal tree for a given full topology. This part was mainly elaborated in Smith [1992] who also presented a C-code. It is essential to implement formula (2) efficiently. Without a highly effective code to find the relatively minimal Steiner tree given a (*n*–2)-component vector that represents a full topology, it would be hopeless to implement a useful procedure to find the minimal Steiner tree. At each iteration from one topology to an neighboring one, about 60*n* linear systems in 2*n*–4 variables to find the best neighbor must be solved; for problems with “only” 30 terminals that is about 1800 systems with 56 unknowns. Based on Smith's highly efficient code, it was feasible to handle such problems on today's 80486-personal computers.

IMPLEMENTATION

The program has been implemented in a highly portable Pascal syntax. The code can be obtained by contacting the author. A small drawing toolbox for displaying Steiner trees on the screen has been added too. The package has been written for educational purposes and not for speed. Therefore, highly readable modules and data structure have been implemented for full topologies, for the tabu-list and for the Steiner tree itself. Anyway, I am convinced that speed does not come from a cryptically written assembler code but from the "right" algorithms and clearly designed structuring of the code.

The main procedures in the package are: *BuildTreeFromTopology* which creates the Steiner data structure from a full topology; *OptimizeTree* which finds the relatively minimal tree; *InitialSolution* which finds a initial solution based on the minimal spanning tree; *BestNeighbor* which finds the best neighboring topology; and *TabuSearchSteiner* which is the main local search procedure with a tabu-search based escape function.

Several procedures have been implemented for the Torricelli construction as well (Steiner tree with three points).

```
{------ primitive , non-compatible draw and other functions }
 procedure DoRandomize;     { initialize the random seed }
 function RandINT (n: integer): integer; { a INT random number }

 procedure Init_Graph;   { initializes the graph port }
 procedure End_Graph;    { shut down the graph port }
 procedure ClearPort;    { empty the graph port }
 procedure DrawStringAt (x, y: integer; a: string);
 procedure DrawIntAt (x, y: integer; a: longint);
 procedure DrawRealAt (x, y: integer; a: real);
 procedure DrawAPoint (p: MyPoint);
 procedure DrawAPoint1 (p: MyPoint);
 procedure DrawAsmallPoint (p: MyPoint);
 procedure DrawALine (p1, p2: MyPoint);

{--- compatibles drawing functions and others ---}
 function Slope (p1, p2: MyPoint): real;
 procedure DrawASegment (se: segment);
 procedure GenerateAPointSet (var po: PointSet; n: integer);
 procedure GenerateAPoint (var p: MyPoint);
 function Distance (p1, p2: MyPoint): real;
 function ccw (p0, p1, p2: MyPoint): integer;
{---- minimum spanning tree , an O(n^2) algorithm ----}
 procedure MST (var po: PointSet; var M: EdgeList);
 procedure DrawMST (var M: EdgeList; labeled: integer);
```

```
   {------ full topologies , data structure -----}
    type  FullTopology = array[0..NMAX] of integer;
    procedure GenerateFirstTopology (var T: FullTopology; n: integer);
    procedure GenerateNextTopology (var T: FullTopology; n: integer);
    procedure GenerateNextTopologyDown (var T: FullTopology; n: integer);
    procedure GenerateRandomTopology (var T: FullTopology; n: integer);
    procedure ShakeTopology (var T: FullTopology; n: integer);
    procedure copyTopology (var T1, T2: FullTopology; n: integer);
    procedure writeTopology (var T: FullTopology; n: integer);

   {------ tabu list , data structure as circular queue -------}
    const  MAXQ = 50;
    type
     TTabu = record
       len: integer;   head, tail: integer;
       Q: array[1..100] of integer;    {----circular queue}
      end;

    procedure InitTabu (var Ta: TTabu; len: integer);
    procedure ClearTabu (var Ta: TTabu);
    procedure AddToTabu (var Ta: TTabu; E: integer);
    function IsInTabu (var Ta: TTabu; E: integer): boolean;

   {---- steiner tree data structure -----}
    type
     TSteiner = record
       n, k: integer;                              { number of terminals , k=n-3
}
       p: PointSet;                            { n terminals + n-2 Steiner points
}
       adj: array[0..NMAX, 0..2] of integer;  { adjacence matrix (for a
topology)}
                       { adj[2,0]=3, adj[2,1]=4 , and adj[2,2]=0 }
                       { means that Steiner point 2 is adjacent to point 3 and 4
}
       edge: array[0..NMAX] of record         {edges list of }
         s, d: integer;
        end;
      end;

    procedure GenerateASteiner (var S: TSteiner; n: integer);
    procedure GenerateSteinerGrid (var S: TSteiner; st: string);
    function TreeLength (var S: TSteiner): real;
    procedure DrawATree (var S: TSteiner; labeled: integer);
    procedure writeSteiner (var S: TSteiner);
    procedure BuildTreeFromTopology (var S: TSteiner; var T: FullTopology);
    procedure OptimizeTree (var S: TSteiner);
    procedure OptimizeTree1 (var S: TSteiner);
    procedure InitialSolution (var S: TSteiner; var T: FullTopology);
    function BestNeighbor (var S:TSteiner; var T:FullTopology; var Ta:TTabu):
real;
    procedure TabuSearchSteiner (S: TSteiner);
```

CONCLUSION

I have presented a general local search heuristic for the Euclidean Steiner tree problem. The heuristic has four parts: an initial solution procedure, a local search procedure, a tabu-search based escape procedure, and an optimizing procedure to find the relatively minimal tree. The first procedure is very efficient to find a good Steiner tree, and is comparable to other heuristics [Smith/Lee/Liebman 1981, Chang 1972]. Part two and three need to be further explored. The implementation is not so convincing, mainly because the implemented neighborhood structure seems not to be of great interest for bigger problems. Nevertheless, the local search finds usually some better solutions for an already good starting solution. Part four was already established by Smith [1992] and there are little to say here. Smith uses the code to implement a branch-and-bound algorithm for the Steiner tree problem and can solve problems with 12 terminals to optimality. This is excellent, if we compare this with highly specialized code for the ESP.

The presented heuristics also has the advantage that it is independent of the metrics as long as it is convex. This is not the case for the most implementations.

A further objective of the paper was to present a connection between the numbers of full topologies in the ESP and the second-order Eulerian numbers. The enumeration scheme might give some insight on a “better” neighborhood structure, but further research are needed to fully take advantage of these connections.

Acknowledgments*:* This paper presents an implementation that was extensively influenced and stimulated by the profound paper of Smith W.D. [1992]. I would also like to thank Daniel Raemy for reading the paper. Of course, I'll have to answer for any remaining errors.

REFERENCES

BERN M.W., GRAHAM R.L., [1989], The Shortest-Network Problem, in: Scientific American, January 1989, p 66–71.

CHANG S-K., [1972], The Generation of Minimal Trees with a Steiner Topology, in: Journal of the ACM, Vol. 19, p 699–711.

CLARK R.C., [1981], Communication networks, soap films and vectors, in: Phys. Educ. 16, p 32–37.

COCKAYNE E.J [1970], On the Efficiency of the Algorithm for Steiner Minimal Trees, in: SIAM Journal Appl. Math., Vol. 18, No. 1, p 150–159.

COCKAYNE E.J., HEWGILL D.E., [1992], Improved Computation of Plane Steiner Minimal Trees, in: Algorithmica, Vol. 7, p 219–229.

COCKAYNE E.J., HEWGILL D.E., [1986], Exact Computation of Steiner Minimal Trees in the Plane, Imformation Processing Letters, Vol. 22, p 151–156.

COCKAYNE E.J., SCHILLER D.G., [1972], Computation of Steiner minimal trees, in: Welsh D.J.A., Woodall D.R. (eds), Combinatorics, Proceedings, The Institute of Mathematics and its Applications, Southend-on-Sea, Essex, p 53–71.

COURANT R., ROBBINS H., [1941], What is Mathematics?, Oxford University Press, New York.

DU D.Z., HWANG F.K., [1990], The Steiner ratio conjecture of Gilbert and Pollak is true, in: Proc. Natl. Acad. Sci., Vol. 8, p 9464–9466.

FOULDS L.R., [1986], Testing the Theory of Evolution: A Novel Application of Combinatorial Optimization, in: Discrete Applied Mathematics, Vol. 15, p 271–282.

GALLAWA R.L., [1978], Application of the Steiner Street Problem to a More Economical Data Bus, in: Fiber and Integrated Optics, Vol 1, No. 3, p 289–311.

GARDNER M., [1986], Mathematical Games, in: Scientific American, June 1986, p 14–17.

GAREY M.R., GRAHAM R.L., JOHNSON D.S., [1977], The complexity of computing Steiner minimal trees, in: SIAM J. Appl. Math., Vol. 34, No. 4, p 835–859.

GAREY M.R., JOHNSON D.S., [1977], The rectilinear Steiner tree problem is NP-complete, in: SIAM J. Appl. Math., Vol. 32, No. 4, p 826–834.

GEORGAKOPOULOS G., PAPADIMITRIOU C.H., [1987], The 1-Steiner Tree Problem, in: Journal of Algorithms Vol. 8, p 122–130.

GILBERT E.N., POLLAK H.O., [1968], Steiner Minimal Trees, in: SIAM J. Appl. Math., Vol. 16, No. 1, p 1–29.

GLOVER F., [1989], Tabu Search – Part I, in: ORSA Journal on Computing, Vol. 1, No. 3, Summer 1989, p 190–206.

GLOVER F., [1990], Tabu Search – Part II, in: ORSA Journal on Computing, Vol. 2, No. 1, Winter 1990, p 4–32.

GRAHAM R.L., KNUTH D.E., PATASHNIK O., [1989], Concrete Mathematics, A Foundation for Computer Science, Addison–Wesley Publ., Reading, Massachusetts.

HWANG F.K., RICHARDS D.S., [1992], Steiner Tree Problems, in: Networks, Vol. 22, p 55–89.

HWANG F.K, RICHARDS D.S., WINTER P., [1992], The Steiner Tree Problem, Monograph, Annals of Discrete Mathematics, Vol. 53, North-Holland.

HWANG K.F., [1986], A linear time algorithm for full Steiner trees, in: Oper. Res. Lett. Vol. 5, p 235–237.

HWANG F.K., SONG G.D., TING G.Y., DU D.Z., [1988], A Decomposition Theorem on Euclidean Steiner Minimal Trees, in: Discrete Comput. Geom., Vol. 3, p 367–382.

KHOURY B.N., PARDALOS P.M., [1993], A Test Problem Generator for the Steiner Problem in Graphs, in: ACM Transaction on Mathematical Software, Vol. 19, No. 4, December 1993, p 509–522.

KUHN H.W., [1974], Steiner's-problem revisted, in: Dantzig G.B., Eaves B.C. (eds), Studies in Optimization, Studies in Math. (10), Math. Assoc. Amer., p 52–70.

MACULAN N., [1987], The Steiner Problem in Graphs, in: Ann. Discrete Math 31, p 185–222.

MELZAK Z.A., [1961], On the problem of Steiner, in: Canadian Mathematical Bulletin, Vol. 4, No. 2, p 143–148.

MIEHLE W., [1958], Link-length minimization in networks, in: Operations

Research, Vol. 6, p 232–243.

POLYA G., [1945], How to Solve it, Penguin Books (renewed second edition 1990, foreworded by Ian Stewart).

POLYA G., [1954], Mathematics and Plausible Reasoning, Vol. 1, Induction and Analogy in Mathematics, Vol II, Patterns of Plausible Inference, Princeton University Press, Princeton, New Jersey (Twelfth printing 1990).

SMITH J.M., LEE D.T., LIEBMAN J.S., [1981], An O(n log n) Heuristic for Steiner Minimal Tree Problems on the Euclidean Metric, in: Networks, Vol. 11, p 23–39.

SMITH W.D. [1992], How To Find Steiner Minimal Trees in Euclidean d-Space, in: Algorithmica, Vol. 7, p 137–177.

SOUKUP J., [1977], Minimum Steiner trees, roots of a polynomial, and other magic, in: SIGMAG/ACM-Bulletin, No. 22, Dezember, 1977, p 37–51.

TRIETSCH D., HWANG F., [1990], An Improved Algorithm For Steiner Trees, in: SIAM Journal Appl. Math., Vol. 50, No.1, p 244–263.

VOSS S., [1990], Steiner-Probleme in Graphen, Mathematical Systems in Economics, Vol. 120, Anton Hain, Frankfurt a.M.

WINTER P. [1987], Steiner Trees in Networks: A Survey, in: Networks, Vol. 17, p 129–167.

WINTER P, [1985], An Algorithm for the Steiner Problem in the Euclidean Plane, in: Networks, Vol. 15, p 323–345.

GO II

Second International Colloquium on Graphs and Optimization

Loèche-Les-Bains, Switzerland

August 21–25, 1994

A tabu-search based local search heuristic for the Euclidean Steiner tree problem

by

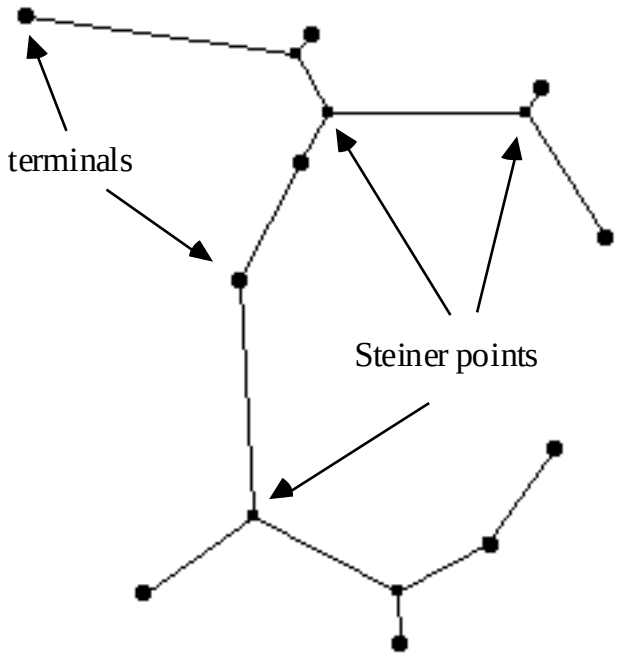
Tony Hürlimann

IIUF (Institute for Informatics)

University Fribourg, Switzerland

The Steiner tree problem

- **A set of points is given (terminals)**
- **A set of points is to be found (Steiner points) such that the tree-length is minimal**



Objectives

- Brief history of the problem and the solution procedures.
- Some facts about Steiner trees.
- Connection about the Eulerian numbers and the numbers of full topologies in the Steiner tree problem.
- Resulting enumeration scheme for a neighborhood structure in a local general heuristic.
- Implementation and results of the heuristic.

Some historical highlights

- **Fermat's problem and Torricelli's construction.**
- **The generalized problem.**
- **The Euclidean Steiner tree problem (Courant & Robbins).**
- **Melzak's algorithm and its variants.**
- **Heuristics based on the minimal spanning tree.**

Definitions

- **Given n points (terminals) find the shortest network connecting the n points.**

- **The shortest network is a tree.**
- **The shortest network contains at most n–2 Steiner points.**
- **Each Steiner point has exactly three adjagent edges, each terminal has at most three adjacent edges.**
- **Any angle between edges is at least of degree 120°.**

- **It follows: the number of topologies is finite:**
  - **all with n–2 Steiner points (full topologies)**
  - **all with n–3 Steiner points**
  - **. . . .**
  - **all with 0 Steiner points.**

- **Another problem is: given a topology, find the Steiner points.**

Two problems to solve.

- **There is no need to enumerate all topologies. Enumerate all full topologies is enough!**

- **Given a topology, there exists an O(n) algorithm to find the Steiner points.**

Full topologies, Eulerian numbers

– **There are: 1·3·5· ... ·(2n–5) full topologies).**

n	3	4	5	6	7	8	9	10	11
$f(n)$	1	3	15	105	945	10,395	135,135	2,027,025	34,459,425

– **They can be represented by a (n–3) component vector, whose i-th entry a_i is an integer $0 \le a_i \le 2i$.**

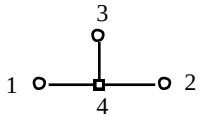
The 15 full topologies of n=5 problems are represented by:

00, 01, 02, 03, 04, 05, 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 24, 25.

– **The number of full topologies of n+2 terminals is the same as the sum of all (second-order) Eulerian numbers of rank n:**

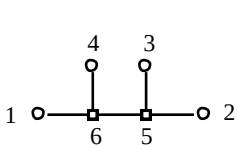
$$\sum_k \left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle = (2n-1)(2n-3)\mathrm{K} \quad (1)$$

Full topologies with n=3,4,5

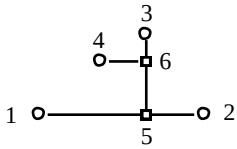


11

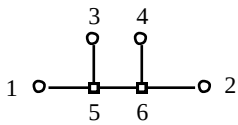
n=3



1122

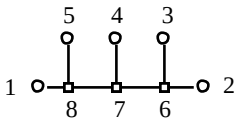


1221

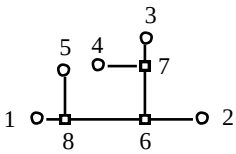


2211

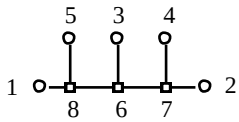
n=4



112233

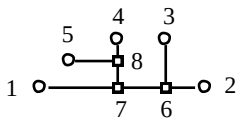


122133

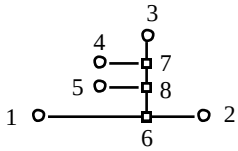


221133

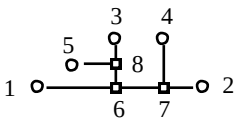
n=5



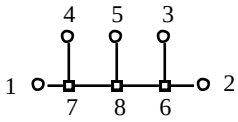
112332



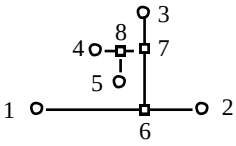
122331



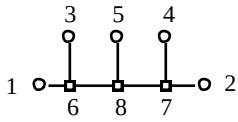
221331



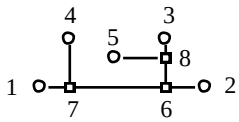
113322



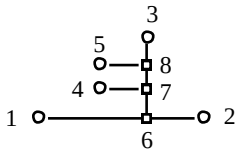
123321



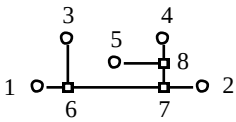
223311



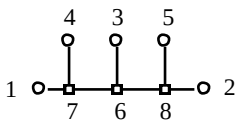
133122



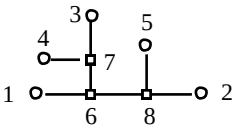
133221



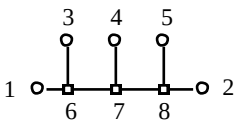
233211



331122



331221



332211

Local search heuristic

1 **find one or several “good” starting solutions, i.e. to find full topologies that produce good relatively minimal Steiner trees.**

2 **determine a efficient neighborhood structure between full topologies for navigate in the solution space of all full topologies effectively.**

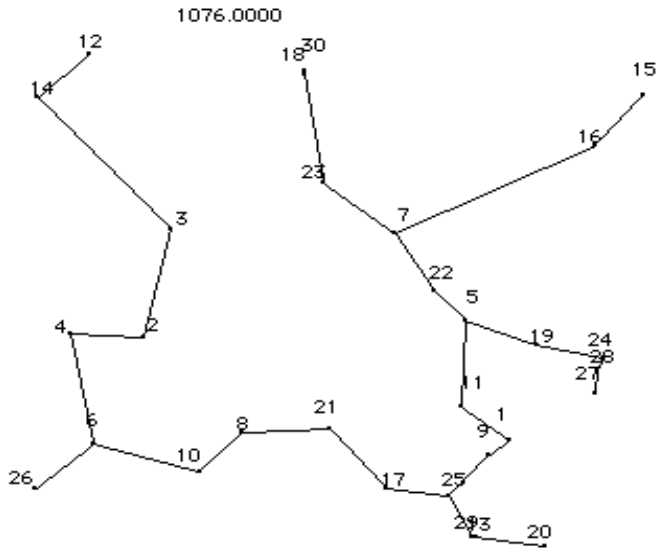
3 **establish a procedure to escape from local minima, such as a particular tabu-search scheme.**

4 **once a full topology, say FT, is given, find efficiently the relatively minimal Steiner tree in D(FT).**

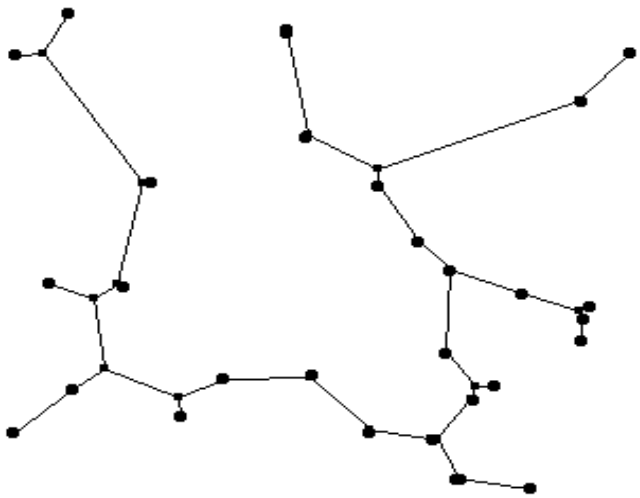
– **Given n points (terminals) find the shortest network connecting the n points.**

Finding a "good" starting solution

- **construct the minimal spanning tree (MST)**

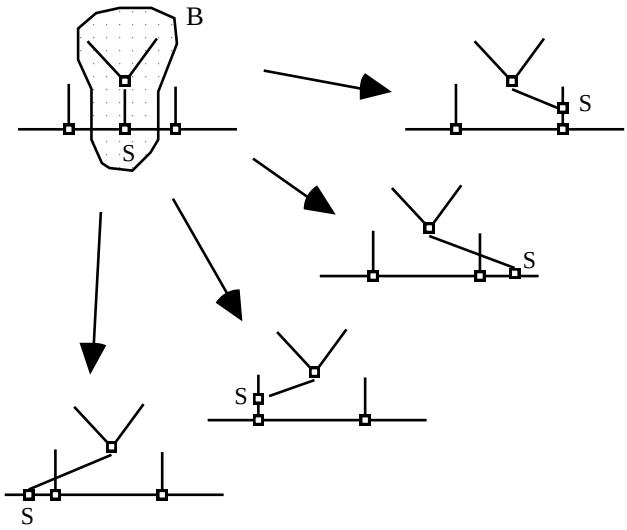


- **produce a full topology "close" to the MST**
- **find the Steiner points to this full topology**



A neighborhood structure

– **for any Steiner point detach one of the three branches and glue it to a adjacent edge**



– **An easier structure is to manipulate directly the (n–3)-component vector: apply the following operation to any component c:**

$$c_i^{new} = (c_i^{old} \pm 2) \bmod (2i)$$

– **A local search method explores repeatedly all neighbors and accepts the best between them.**

A Tabu search scheme

- **Tabu is a (circular) queue of k entries.**
- **The component that led to the best neighbor with a greater minimum than the actual solution will be fixed.**
- **This scheme led (supposedly) to the optimal solution of most problems with up to n=10 terminals.**
- **The scheme was not successful for more points.**

Finding Steiner points – given a topology

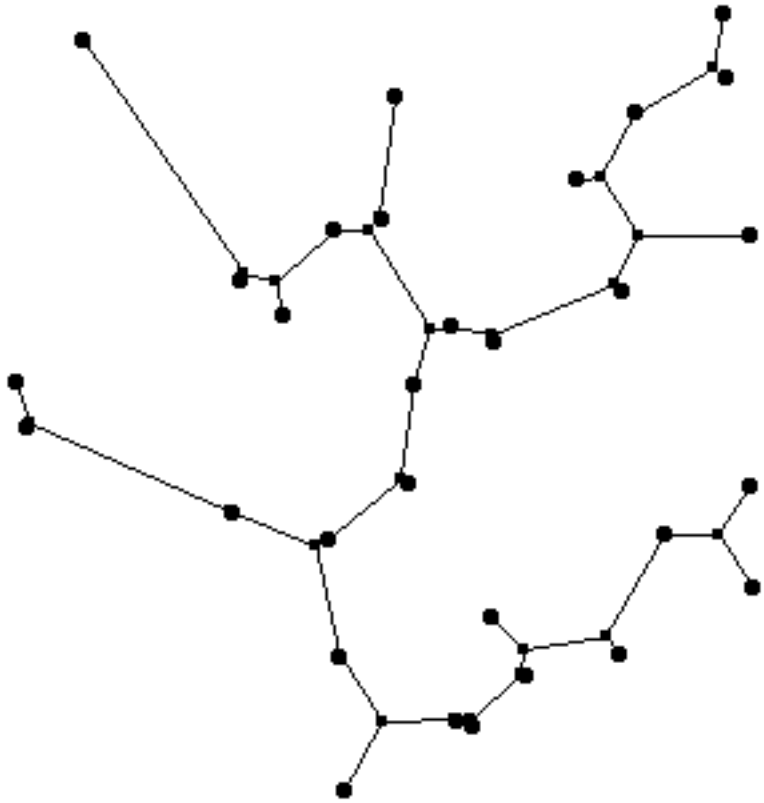
- **The procedure was given in Smith [1992]: solve repeatedly the following linear system**

$$x_k^{i+1} = \frac{\sum_j \frac{x_j^{i+1}}{\Delta_{jk}}}{\sum_j \frac{1}{\Delta_{jk}}} \qquad y_k^{i+1} = \frac{\sum_j \frac{y_j^{i+1}}{\Delta_{jk}}}{\sum_j \frac{1}{\Delta_{jk}}} \qquad (2)$$

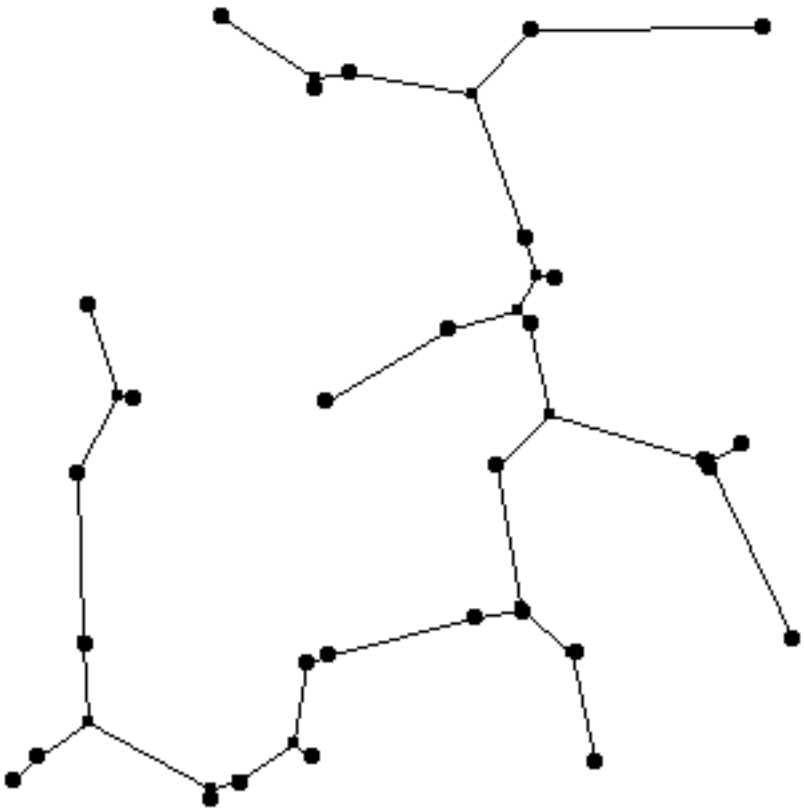
(for n=30, solve about 1800 linear systems with 56 variables).

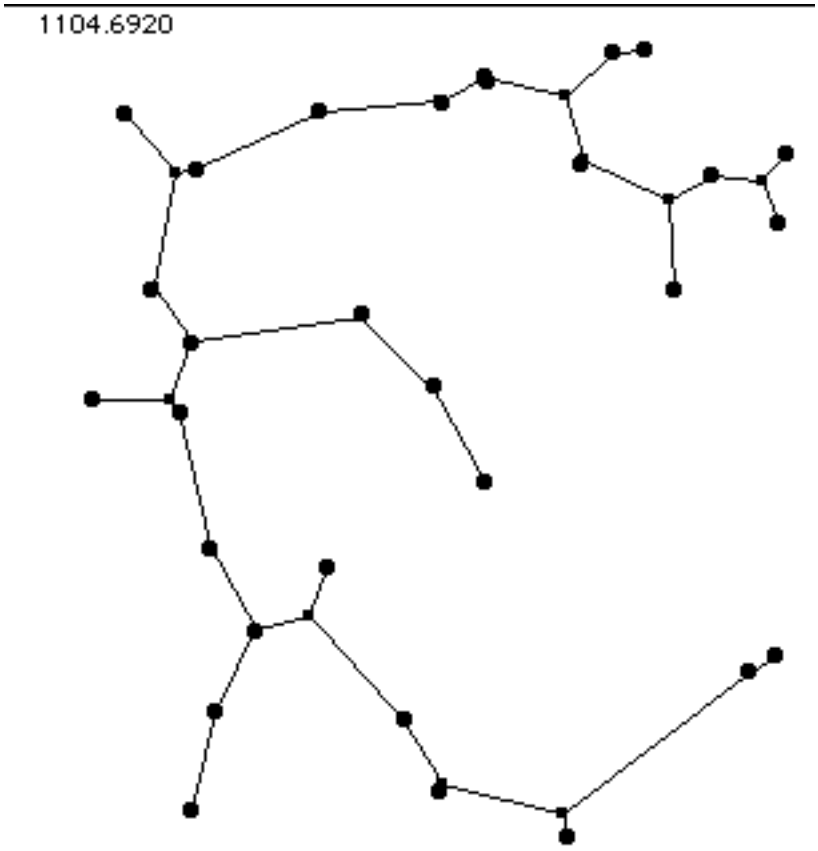
- **There exists a O(n) algorithm, since the linear system is a tridiagonal system.**

1121.2770



1161.1029




1104.6920