
ELSEVIER

Computational Geometry 18 (2001) 55–64

Computational Geometry
Theory and Applications

www. elsevier.nl/locate/comgeo

Enumerating the k best plane spanning trees

Ambros Marzetta, Jurg Nievergelt *

ETH, Informatik, 8092 Zurich, Switzerland

Communicated by K. Mehlhorn; received 9 August 2000; received in revised form 2 November 2000; accepted 13 November 2000

Abstract

A spanning tree constructed of straight line segments over a set of points in the Euclidean plane is called "non-crossing" or "plane tree", if no two segments intersect. Imposing the additional constraint of non-crossing segments makes many combinatorial geometric problems harder. In the case of plane spanning trees, however, we show that they can be enumerated efficiently in the order of their total length. This makes it possible to efficiently find the k best plane trees, or all those shorter than a given bound.

Keywords: Combinatorial optimization; Enumerative optimization; Computational geometry; k best spanning trees; Exhaustive search; Reverse search

1. Geometry, combinatorics and enumerative optimization

Problems about discrete spatial configurations involve a challenging interaction between the continuum characteristic of geometry and the discreteness of combinatorics. Each of these disciplines has evolved its characteristic techniques that exploit the nature of the objects treated, continuous or discrete. When used together to attack a geometric-combinatorial problem, surprises lurk beneath the surface. A seemingly minor change in a geometric specification may clash with combinatorial assumptions, and vice versa. This interaction between geometric and combinatorial constraints makes it difficult to predict, on an intuitive basis, which problems are amenable to efficient algorithms, and which are not.

The difficulties mentioned are exacerbated when we aim at enumerating all spatial configurations that meet certain specifications, not in any arbitrary order convenient for enumeration, but rather in a prescribed order. Search techniques impose their own restrictions on the order in which they traverse a search space, and these may be incompatible with the desired order.

We present an example of a simple geometric-combinatorial search and enumeration problem as an illustration of issues and techniques: the enumeration of plane spanning trees over a given set of points

* Corresponding author.
E-mail address: jn@inf.ethz.ch (J. Nievergelt).

0925-7721/01/$ – see front matter
PII: S0925-7721(00)00029-8

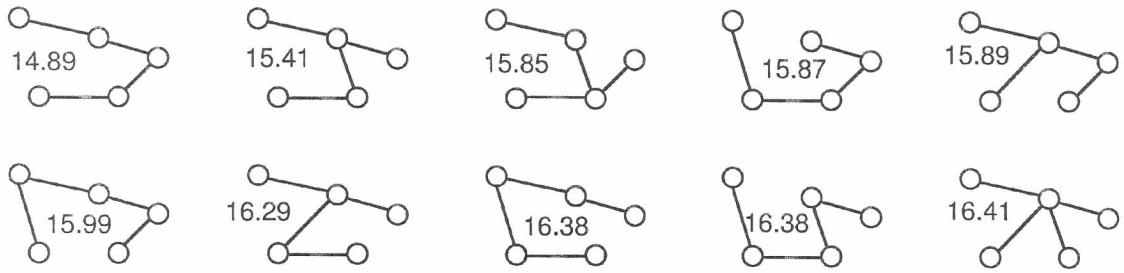

Fig. 1. The 10 shortest plane spanning trees over five given points, in order of increasing length.

in the plane, i.e. those trees constructed with straight line segments in such a manner that no two edges intersect. Avis and Fukuda [1] present an algorithm for enumerating all plane spanning trees, in some uncontrolled order that results from the arbitrary labeling of points. We attack this same problem under the additional constraint that these plane spanning trees are to be enumerated according to their total length (i.e. the sum of the lengths of their edges), from short to long. We may stop the enumeration after having listed the k shortest trees, or all trees shorter than a given bound c. Fig. 1 lists the 10 shortest plane spanning trees among the 55 on the particular configuration of 5 points with coordinates (0, 5), (1, 0), (4, 4), (5, 0), (7, 3).

Algorithms that enumerate all the elements of some discrete set S often proceed by defining a search tree over S. An easily constructed element is designated as the root. An adjacency operator A assigns to every element s a set $A(s)$ of neighbors, usually based on some structural and lexicographic proximity. Any of a number of standard search strategies, such as depth-first, breadth-first, best-first, traverse the tree in different orders, using different data structures to keep track of what has been visited and what remains to be done.

A difficulty arises when the goal is not just to enumerate a set in some arbitrary order, but in a prescribed order. This is the case in enumerative optimization, where the goal is to enumerate the elements of S from best to worst according to some objective function f defined on S. One may wish to stop the enumeration after the k best elements, or after having seen all elements s with $f(s) \leqslant c$.

In recent years the literature dealing with "k best" problems has grown significantly. The techniques used vary widely according to the problem investigated. In particular, Gabow [4], Camerini et al. [3] and Katoh et al. [5] present algorithms for generating the k best spanning trees or arborescences for graphs, but they do not consider the Euclidean case, and hence do not have to deal with any geometric constraint such as "no cross-over". The enumerative optimization algorithm to be presented uses only elementary geometry and standard search techniques.

2. The space of plane spanning trees on a Euclidean graph

Consider a graph $G = (V, E, w)$ with n vertices $p, q, r, \ldots$ in V, weighted edges $e = (p, q)$ in E, and a weight function $w : E \to \mathbb{R}$. The set of spanning trees over G has a well-known useful structure that is exploited by several algorithms, in particular for constructing a minimum spanning tree (MST). The structure is based on an exchange operator, an "edge flip", and on a monotonicity property.

Let T be any spanning tree over G, e' an edge not in T, $\mathrm{Ckt}(e', T)$ the unique path P in T that connects the two endpoints of e', and e any edge in P. The edge flip $T' = T - e + e'$ that deletes e from

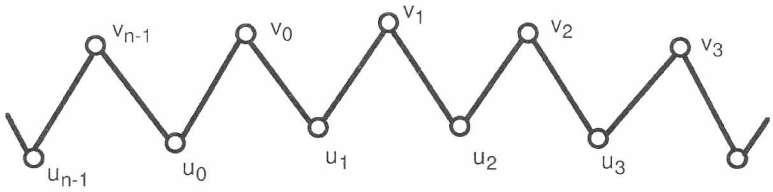

Fig. 2. A sector of the gear wheel of MSTs.

T and replaces it by e' creates a new spanning tree T' that is "adjacent to T". If $w(e) > w(e')$, this edge flip is profitable in the sense that $|T'| < |T|$, where $|T|$ denotes the total length of T. The remarkable fact exploited by algorithms for constructing an MST is that in the space of all spanning trees over G, with edge flip as an adjacency operator, any local minimum is also a global minimum. This implies that any greedy algorithm based on profitable edge flips or on accumulating the cheapest edges (e.g., Prim, Kruskal) converges towards an MST.

In this paper we study *Euclidean* graphs and *plane* spanning trees. A Euclidean graph is a complete graph whose vertices are points in the plane, and whose edge weights are the distances between the endpoints of the edge. For $p, q \in V$ and $e = (p, q)$, let $|(p, q)|$ denote the length of edge e. For a Euclidean graph it is natural to consider "plane" or non-crossing spanning trees, i.e. trees no two of whose edges cross. The triangle inequality implies that any MST over a Euclidean graph is plane, i.e. has no crossing edges.

In the following section, we define a search tree for enumerating all plane spanning trees, in order of increasing total length, over a given point set in the plane. This search tree is presented in such a form that standard tree traversal techniques apply, in particular, reverse search of Avis and Fukuda [1]. Specifically, we define a unique root R which is an MST; and a monotonic gradient function g that assigns to each tree $T \neq R$ a tree $g(T)$ with $|g(T)| \leqslant |T|$. The gradient function g has the property that, for any $T \neq R$, some iterate $g(\ldots, g(T))$ equals R, i.e. R is a sink of g; hence g generates no cycles. For efficiency's sake, both g and its inverse can be computed efficiently.

For simplicity of expression we describe the geometric properties of the search tree as if the configuration of points was non-degenerate in the sense that there is a unique MST, that $|g(T)| < |T|$, that no two edges have equal length, and that no two angles (defined by any triple p, q, r of points) are equal. If this is not the case, and equal quantities must be compared, the tie is broken according to a static (lexicographic) ordering of all the quantities concerned: the numbering of the vertices $1, \ldots, n$ induces a total order on the edges (p, q) and on all the angles (p, q, r).

Unfortunately, the space of *plane* spanning trees over a Euclidean graph does not exhibit as simple a structure as the space of *all* spanning trees. It is evident that if T is a plane tree, an edge flip $T' = T - e + e'$ may introduce cross-overs. When searching for a gradient function g whose only sink is the MST, it is tempting to consider edge flips that introduce MST edges, and one might expect that these restricted flips preserve planeness. The counterexample of Figs. 2 and 3, however, shows a set of points together with a spanning tree none of whose edges can be replaced by an MST edge without introducing cross-overs.

Consider a "gear wheel graph" of $2n$ vertices: n inner vertices $u_0, \ldots, u_{n-1}$ are equally spaced on a circle of some large radius r; n outer vertices $v_0, \ldots, v_{n-1}$ are equally spaced on a slightly larger circle of radius $r + 1$. The parameters n and r are chosen such that $|(u_i, u_{i+1})| > |(u_i, v_i)|$ and $|(u_i, u_{i+1})| > |(v_i, u_{i+1})|$. Thus, any MST consists of $2n - 1$ edges that link an inner node to a neighboring outer node, and looks like a gear wheel that lacks "half a tooth". Specifically, it contains all but one of

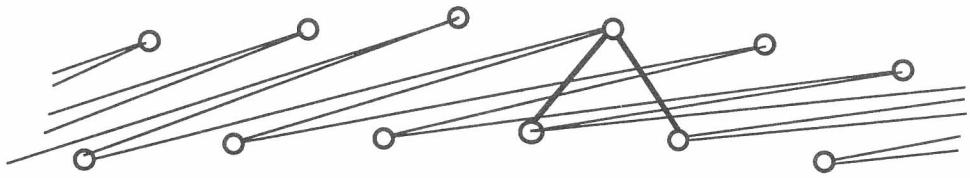

Fig. 3. A skewed gear wheel resists MST edge flips.

the $2n$ edges (u_i, v_i) and (v_i, u_{i+1}), where all indices are taken mod n. Each of the $2n$ edges that form the gear wheel can be part of some MST, and no other edge can.

Fig. 3 shows a sector of a "skewed gear wheel" plane spanning tree T: its edges (u_i, v_{i+3}) and (v_i, u_{i-2}) are drawn as thin lines. Now consider any of the $2n$ edges that can be part of some MST (thick lines). An edge of the form (v_i, u_{i+1}) intersects 3 edges of T; an edge of the form (u_i, v_i) intersects 5 edges of T. Thus, flipping any edge of T for any MST edge introduces cross-overs.

The example of Fig. 3 shows a plane spanning tree T and an edge e not in T such that there exists no e' suitable for an edge flip that avoids crossing edges. Moreover, this is true not just for some unfortunately placed edge e, but for "good" edges, i.e. all edges that can be part of some MST.

The edge flip operator $T' = T - e + e'$ is the key to efficient spanning tree algorithms, but the example shows that this operator applies to plane spanning trees under restrictive conditions only. In order to develop algorithms that search the space of plane spanning trees efficiently, it is of crucial importance to define a class of edge flips that introduce no cross-over.

3. Edge flips that avoid cross-over

The geometric key issue to be solved is an efficient way of finding edge flips that (1) shorten the tree and (2) avoid cross-over. We distinguish two cases, triangle flips and Gabriel graph flips.

3.1. Triangle flips

With the goal of limiting the number of cases that may arise, we consider "triangle flips", a special type of edge flip that involves only three distinct points, rather than four. Consider a tree T containing the edge (p, r), and any third point q. At least one of the edges (q, p), (q, r) is not in T, and hence it is always possible to execute precisely one of the flips $T' = T - (p, r) + (q, p)$ or $T' = T - (p, r) + (q, r)$. In addition, if we choose the triangle (p, q, r) to be empty, i.e. to contain no point of V nor any portion of any edge in T, we can guarantee that such a triangle flip introduces no cross-over.

Specifically, among all point triples (p, q, r) such that (p, r) is in T, select the one whose angle(p, q, r) is maximum. This ensures that the triangle (p, q, r) is empty in the sense that no edge of the current tree T touches its interior. Fig. 4 illustrates the argument that the triangle (p, q, r) must be empty, and hence that this triangle flip does not introduce any crossing edges. At left, consider the possibility of an edge e one of whose endpoints u lies in the triangle (p, q, r). The inequality angle$(p, q, r) <$ angle(p, u, r) contradicts the assumption that angle(p, q, r) is maximum. At right, consider the possibility of an edge $e = (u, v)$ that crosses both (p, q) and (q, r). Then angle$(u, q, v) >$ angle(p, q, r), again a contradiction. Thus, neither (p, q) nor (q, r) cause a cross-over if flipped for (p, r).

Fig. 4. The assumption of new cross-overs contradicts the choice of "angle(p, q, r) is maximum".

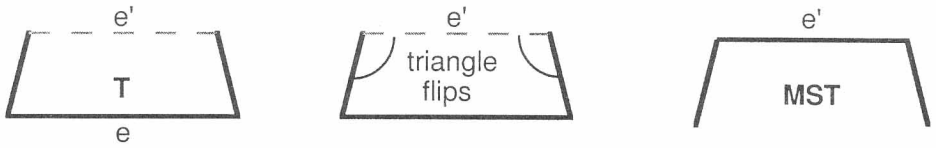

Fig. 5. In the trapezoid above, the spanning tree T is transformed into the MST by flipping edges e, e'. But no sequence of profitable triangle flips achieves this goal.

Whereas triangle flips based on the maximum angle avoid cross-overs, they are insufficient for two reasons:

1. the triangle flip involving the maximum angle is not necessarily profitable, and
2. profitable triangle flips alone are insufficient to transform an arbitrary spanning tree T into an MST, as Fig. 5 shows.

We deal with these two problems as follows.

1. A triangle flip is profitable if the maximum angle is large enough. Among various possible choices, it is convenient to choose "$\geqslant 90°$" as a definition of "large enough". As illustrated in Fig. 4, this definition is equivalent to "the vertex q of the maximum angle(p, q, r) lies in the disk with diameter (p, r)". Thus, if the vertex q of the maximum angle(p, q, r) lies in the disk of the tree edge (p, r), we use it in a non-crossing profitable triangle flip. We formulate this case after introducing the useful concept of the Gabriel graph.
2. Consider any set C of non-crossing edges over the given point set V, and any plane tree T contained in C. Trivially, any edge flip limited to edges in C cannot introduce any cross-over. We seek a set C, a "skeleton of G", that is dense enough so as to contain a sufficient number of spanning trees, and has useful geometric properties. After triangle flips have transformed an arbitrary spanning tree into a subset of this skeleton, we switch to edge flips that lie within C, but are otherwise unrestricted. Among various choices for C, the Gabriel graph of V will do.

3.2. The Gabriel graph as an intermediary between any tree and the MST R

Definition. The Gabriel graph GG(V) over a point set V contains an edge (p, q) iff no point r in $V - \{p, q\}$ is in the closed disk disk(p, q) over the diameter (p, q). The useful geometric properties mentioned include the following:

1. GG(V) has no crossing edges.
2. Consider any point x (not in V) that lies inside disk(p, q). Then dist$(x, p) <$ dist(p, q), dist$(x, q) <$ dist(p, q), angle$(p, x, q) > 90°$.

3. Any MST over V is contained in GG(V). (Proof: consider any edge e of an MST. If there was any point $p \in V$ inside disk(e), e could be exchanged for a shorter edge.)

We can now define the monotonic gradient function g discussed in Section 2. In a first phase, a sequence of triangle flips transforms any spanning tree T into a tree T' which is a subset of GG(V); thereafter, a sequence of edge flips within GG(V) transforms T' into the MST R. The set of overlapping g-trajectories from any T to R defines the search tree needed to enumerate the spanning trees in the order from short to long.

Rule 1. Triangle flips. Consider a tree T not contained in GG(V), and hence not an MST. Among all point triples (p, q, r) such that (p, r) is in T, select the one whose angle(p, q, r) is maximum. The properties of the Gabriel graph imply the following assertions: angle$(p, q, r) \geqslant 90°$, (p, r) is not in GG(V), $|(q, p)| < |(p, r)|$ and $|(q, r)| < |(p, r)|$.

With the notation above, define $g(T) = T - (p, r) + e'$, where e' is either (q, p) or (q, r), chosen such that $g(T)$ is a spanning tree. As shown in Section 3.1, $g(T)$ is a plane tree with $|g(T)| < |T|$.

Rule 2. Flipping edges in the Gabriel graph. Let T be a spanning tree over V contained in GG(V) and different from the (uniquely defined) MST R. Define $g(T) = T - e + e'$, where e' is the lexicographically first edge of R not in T, and e is the longest edge in Ckt(e', T). Obviously, $g(T)$ is closer to R than T is, and if the MST is unique, then $|g(T)| < |T|$.

3.3. Degenerate cases

The two rules described so far work for a point set V which is in general position in the sense described in Section 2. Minor changes suffice to handle the three possible degeneracies that affect Rules 1 and 2:

- Rule 1: The maximum angle(p, q, r) is not unique. Select the lexicographically first triple (p, q, r) with maximum angle.
- Rule 2: The minimum spanning tree is not unique. Define the lexicographically first MST to be R. Despite the possibility that $|g(T)| = |T|$, when both T and $g(T)$ are MSTs, the iteration of g does not generate cycles because $|g(T) \cap R| > |T \cap R|$.
- Rule 2: The longest edge in Ckt(e', T) is not unique. Select the lexicographically first longest edge as e.

4. Complexity

The general scheme of reverse search and the gradient function g defined in Section 3 together specify an algorithm for enumerating the k best plane spanning trees. We analyze the complexity of this algorithm by distinguishing two types of operations:

a. operations defined at the level of the objects involved, such as computing the parent $g(T)$ of a spanning tree T, or its children, by means of the inverse gradient $g^{-1}(T)$, and
b. more elementary operations that depend on the data structures used to represent these objects.

We discuss these aspects separately in the following sections.

4.1. Fan-out of the search tree

The complexity of tree traversal algorithms depends primarily on two operations: for every node T, repeatedly compute (1) the parent $g(T)$, and (2) all its children, $g^{-1}(T)$. Clearly, the number of a node's children is an inherent component of this algorithm's complexity. The following examples show that, in the worst case, a node can have $\Theta(n^2)$ children, both with respect to Rules 1 and 2.

Fig. 6 shows n points aligned along a quadrant of an ellipse. The unique MST R shown at left has edges that coincide with the curved segments between a pair of points adjacent along the ellipse. The Gabriel graph of these n points is identical with R. At right we see a tree T with a single non-MST edge $e = (p, r)$. In compensation, T lacks the MST-edge $e' = (p, q)$ incident to that endpoint p of e which lies on that side of the ellipse where the curvature is least. Since T is not contained in GG(V), Rule 1 applies with the result $g(T) = R$. Since the edge e connects any pair (p, q) of non-adjacent points, there are $\Theta(n^2)$ such edges e and hence $\Theta(n^2)$ children of R.

Fig. 7 shows $2n$ points arranged at equal intervals along two almost horizontal, almost parallel lines. Additional edges zig-zag between upper and lower points. Distances are chosen such that the unique MST R is as shown in bold at left. Notice that the entire graph is its own Gabriel graph. At right we see a tree T that has a single non-MST edge e, and in compensation lacks an MST edge e'. Since T is contained in GG(V), Rule 2 applies with the result $g(T) = R$. The edge e connects any adjacent pair (p, q) consisting of an upper and a lower point; e' is any MST edge to the left of e. Thus, there are $\Theta(n^2)$ such pairs e, e' and hence $\Theta(n^2)$ children of R.

The fact that a typical node of the search tree can have $\Theta(n^2)$ children strongly suggests that $\Theta(kn^2)$ is a lower bound on the complexity of computing the k best spanning trees. In view of the fact that the statement above is based purely on a counting argument and ignores any data structure operations that are surely needed, one expects a complexity exceeding $\Theta(kn^2)$.

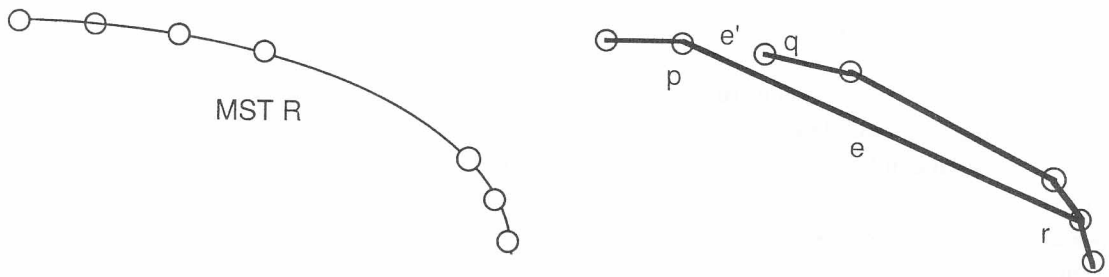

Fig. 6. A graph with nodes along a line of monotonic curvature.

Fig. 7. A ladder graph.

4.2. Computing the children

The complexity of computing the $O(n^2)$ children of a node depends on the data structures chosen to represent the current spanning tree and the state of the search. Our implementation (see also Section 4.4) uses straightforward data structures suited to the demands of simplicity. It is conceivable that a fancier approach may lead to an asymptotic improvement in time complexity, but we doubt that this would be justified in practice. A spanning tree is stored as a list of edges and a (redundant) adjacency list for every vertex. For every edge (p, r), we store the point q which sees (p, r) under the largest angle.

We present the enumeration algorithm as a tree traversal. Note that the tree being traversed is not a spanning tree, but a search tree consisting of nodes which are spanning trees over V. The spanning trees themselves consist of vertices and edges. As the search tree is directed by the gradient function g, we speak of the parent and the children of a node s. For a node (spanning tree) s, we denote its parent by $g(s)$ and its children by t_i. The size of the search tree to be traversed, i.e. the number of spanning trees to be produced, is denoted by k.

The complexity of most tree traversal algorithms depends on two operations: for every node, we have to compute (1) all children, and (2) the parent. We first show that the gradient function which computes the parent is $O(n)$. Then, for every node s we define $O(n^2)$ "candidate children" u_i, a superset of the children of s which can be enumerated efficiently. The algorithm computes the children of a node s in time $O(n^3)$ by enumerating the candidate children and then checking for each candidate u_i whether $g(u_i) = s$. This method leads to a time complexity of $O(kn^3)$ for a straightforward traversal that visits k nodes of the search tree.

The gradient function calls each of the following four operations a (small) constant number of times:

- find an edge (p, r) of the current spanning tree and a point q so that angle(p, q, r) is maximum,
- intersect two spanning trees,
- traverse the unique path between two vertices of a spanning tree,
- replace (flip) an edge of a spanning tree.

Every operation can be performed in linear time on the data structure chosen because a tree has only $n - 1$ edges. Therefore, the gradient function is $O(n)$.

Now we define the candidate children as a superset of the children. A candidate child u_i of s is a spanning tree which can be reached from s by an edge flip which excludes any edge e' of s and includes either (1) an edge of the Gabriel graph or (2) any edge incident to e'. Case 1 includes all children of s to which Rule 1 applies, and case 2 includes all children of s to which Rule 2 applies. In either case, there is a choice of $n - 1$ edges for e' and a choice of $n - 1$ edges to be included. Therefore, there are $O(n^2)$ candidate children.

For every candidate child u_i, we check in linear time that there is no intersection with the included edge and that $g(u_i) = s$. The children t_i of s are those candidates that pass the test. Therefore, the computation of all children of a spanning tree takes time $O(n^3)$, and this is the dominant term in the whole complexity.

4.3. Traversing the search tree

We consider two versions of the problem, and several slightly different algorithms for solving them.

Problem A. Given a set of points and a constant c, enumerate all spanning trees of length $\leqslant c$.

Problem B. Given a set of points and an integer k, enumerate the k smallest spanning trees.

Algorithm A (Depth-first traversal). We traverse the search tree depth-first and prune it at any spanning tree of length $> c$. This is the conventional reverse search of Avis and Fukuda, which takes time $O(kn^3)$ and space $O(n)$, if k elements are output. As a side effect useful for Algorithm B3, we can additionally get the smallest spanning tree $> c$.

Problem B is harder because the search tree can be pruned only when the length of the kth spanning tree is known. We propose three algorithms with different space–time tradeoffs.

Algorithm B1 (Best-first traversal). We maintain (1) the j trees output so far and (2) a priority queue containing, for each of these trees, the best child that does not belong to the j best trees. This data takes space $O(kn)$. When the $(j+1)$th tree is selected from the queue, we need to find its cheapest child, and for the parent node we need to find the next child (with respect to cost). In either case, we can compute all children in time $O(n^3)$ and keep the appropriate one. We conclude that Algorithm B1 takes time $O(kn^3)$ and space $O(kn)$.

Algorithm B2 (Differential encoding). The data structure of Algorithm B2 can be modified to take less space. The MST is represented explicitly, all other trees are encoded concisely: we only maintain a pointer to the parent and a description of the corresponding edge flip. This modification reduces the space requirement to $O(n+k)$. When the $(j+1)$th tree is selected from the queue, we compute the explicit representation of the tree and its parent. These k edge flips take time $O(kn)$. Then, we proceed as in Algorithm B1. Algorithm B2 therefore takes time $O(k^2n + kn^3)$ and space $O(n+k)$.

Algorithm B3 (Iterative deepening). The k best spanning trees can be enumerated in space $O(n)$ by replacing a best-first traversal by an iterative deepening depth-first traversal known from game tree evaluation. This approach performs k depth-first traversals. In the jth traversal, it computes the jth best spanning tree by calling Algorithm A with c equal to the length of the $(j-1)$th best spanning tree. This algorithm takes time $O(k^2n^3)$ and space $O(n)$.

4.4. Benchmark measurements

We have implemented Algorithms A and B3 in C using the parallel search library ZRAM [2,6]. The program is available at http://wwwjn.inf.ethz.ch/ambros/zram/ and outputs a drawing (in PostScript) of the k best plane spanning trees for a given set of points.

The statistics of benchmarks using our implementation presented in Table 1 confirms the asymptotic analysis to a high degree. The running time in seconds was measured on $n = 30$ and $n = 60$ points randomly distributed in the unit square, using a Digital DEC 3000 Model 700 built in 1995.

The table shows the running times growing as $O(kn^3)$ for A and $O(k^2n^3)$ for B3. Algorithm A (input: bound on the length) has the theoretical advantage over B3 (input: number k of trees) by a factor of k. The measurements show a ratio (time B3)/(time A) slightly larger than k.

Table 1

n	k	Time A	Time B3	B3/A
30	20	0.64	14.7	22.9
30	40	1.3	56.3	43.3
30	80	2.6	228	87.7
60	20	4.5	104	23.1
60	40	9.0	411	45.7
60	80	17.0	1574	92.6

5. Conclusion

Any problem defined on graphs can be posed for the special case of Euclidean graphs. If the solution involves a set of edges, in the Euclidean case it is natural to impose the additional constraint of non-crossing edges. It would be of interest to investigate in some generality what effect this additional constraint has on a variety of combinatorial problems.

There are problems where the non-crossing property poses no new constraints at all, because the desired solution automatically excludes cross-overs. This is the case for the Euclidean Traveling Salesman Problem and the Euclidean MST. If we consider the "k-best" version of these two problems, on the other hand, the situation is unclear. For the k-best MSTs we have shown how the search can be confined to the subspace of non-crossing spanning trees. For the k-best Traveling Salesman Tours we do not expect to find an equally simple structure that would lead from tour to tour in a monotonic order. It would be of interest to characterize and distinguish the types of problems that can be solved under the additional constraint of non-crossing edges with and without significant loss of efficiency.

Acknowledgements

Thanks to Kurt Mehlhorn for insightful comments that improved the paper.

References

[1] D. Avis, K. Fukuda, Reverse search for enumeration, Discrete Appl. Math. 65 (1996) 21–46.
[2] A. Brüngger, A. Marzetta, K. Fukuda, J. Nievergelt, The parallel search bench ZRAM and its applications, Ann. Oper. Res., Special Issue on Parallel Optimization 90 (1999) 45–63.
[3] P.M. Camerini, L. Fratta, F. Maffioli, The K best spanning arborescences of a network, Networks 10 (2) (1980) 91–109.
[4] H.N. Gabow, Two algorithms for generating weighted spanning trees in order, SIAM J. Comput. 6 (1977) 139–150.
[5] N. Katoh, T. Ibaraki, H. Mine, Algorithms for finding k minimum spanning trees, SIAM J. Comput. 10 (1981) 247–255.
[6] A. Marzetta, ZRAM: A library of parallel search algorithms and its use in enumeration and combinatorial optimization, Dissertation, ETH Zurich, 1998.