SERIE B — INFORMATIK

Maximum k-Chains in Planar Point Sets: Combinatorial Structure and Algorithms

Stefan Felsner*†
Lorenz Wernisch**‡

B 92–27
December 1992

Abstract

A chain of a set P of n points in the plane is a chain of the dominance order on P. A k-chain is a subset C of P that can be covered by k chains. A k-chain C is a *maximum k-chain* if no other k-chain contains more elements than C. This paper deals with the problem of finding a maximum k-chain of P.
Recently, Sarrafzadeh, Lou, and Lee [SLL90, SL92] proposed algorithms to compute maximum 2- and 3-chains in optimal time $O(n \log n)$. Using the skeleton $S(P)$ of a point set P introduced by Viennot [Vie77, Vie84] we describe a fairly simple algorithm that computes maximum k-chains in time $O(kn \log n)$ using $O(kn)$ space. If our theorems on skeletons are added to Viennot's results, they allow to derive the full Greene-Kleitman theory for permutations from properties of skeletons.

*Technische Universität Berlin, Fachbereich Mathematik, MA 6-1, Straße des 17. Juni 136, W-1000 Berlin 12, Germany, e-mail: felsner@math.tu-berlin.de.
†Stefan Felsner has been supported by the Deutsche Forschungsgemeinschaft under grant FE 340/2-1.
*Freie Universität Berlin, Fachbereich Mathematik, Institut für Informatik, Arnimallee 2-6, W-1000 Berlin 33, Germany, e-mail: wernisch@tcs.fu-berlin.de.
‡Lorenz Wernisch has been supported by a grant from the German-Israeli Binational Science Foundation and by the ESPRIT Basic Research Action No. 7141 (ALCOM II).

1 Introduction

We define an order on a set of n points P in the plane by $p = (x, y) \leq q = (x', y')$ if $x \leq x'$ and $y \leq y'$. A set C of elements of P is a *chain* if any two members p, q of C are comparable, i.e., either $p \leq q$ or $q \leq p$. On the other hand, a set $A \subseteq P$ with no two different points comparable is an *antichain*. If a subset C of P can be covered by k chains it is called a *k-chain*. A k-chain C is *maximum* if no other k-chain contains more elements than C. This paper deals with the problem of finding such k-chains in P. Note that the "greedy method" that repeatedly removes maximum chains may fail in computing a maximum k-chain even for $k = 2$.

Sarrafzadeh, Lou, and Lee [SLL90] propose an algorithm to compute 2-chains in optimal time $O(n \log n)$. Recently, they showed how to find 3-chains within the same time bound [SL92]. They are motivated to consider k-chains by problems in VLSI design, e.g., multi-layered via minimization for two-sided channels. Maximum k-chains also turn out to be useful in computational geometry, e.g., for counting points in triangles (see [MW92]). Futhermore, a permutation $\sigma\colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ may be represented by points $(i, \sigma(i))$ in the plane. Chains and antichains then correspond to increasing and decreasing subsequences and finding maximum k-chains amounts to computing maximum k increasing subsequences.

We describe a fairly simple method to find maximum k-chains for arbitrary k in time $O(kn \log n)$ and with $O(kn)$ space. Our approach is based on the useful concept of the *skeleton* of P introduced by Viennot [Vie84] (see also [Vie77]). We use a maximum $(k-1)$-chain in the skeleton to partition the plane in k appropriate regions. Taking k maximum chains from each region then already yields a maximum k-chain. In [Vie77] Viennot gives some hints how to find k-chains in time $O(n^2 \log n/k)$ and $O(n^2/k)$ space, which is suitable if k is large. Our method now leads to a kind of complementary algorithm.

We want to indicate how our results complete those of Viennot such that the full Greene-Kleitman theory for permutations can now be derived solely from the geometry of skeletons. The reader mainly interested in the algorithmic part of the construction of k-chains may skip the rest of this section and continue reading in Section 2, where definitions and first examples are given. Then, in Section 3, the combinatorial background for the algorithm is developed. Finally, in Section 4, the algorithm is described in fairly detailed pseudocode and the running time is analyzed.

Greene and Kleitman [GK76,Gre76] discovered that the k-chains and ℓ-antichains of an arbitrary partially ordered set P contain a lot of structure. From this theory we quote a theorem relating maximum k-chains to maximum ℓ-antichains.

Theorem 1 (Greene) *To an order P with n elements there exists a partition α of n, such that the Ferrers diagram F_α of α has the following properties:*

(1) *The number of squares in the k longest rows of F_α equals the size of a maximum k-chain, for $1 \leq k \leq n$.*

(2) *The number of squares in the* ℓ *longest columns of* F_α *equals the size of a maximum* ℓ*-antichain, for* $1 \leq \ell \leq n$.

Viennot's construction [Vie84] of a skeleton leads to a geometric interpretation of the well known bijective correspondence between permutations and pairs of Young tableaux (the Robinson-Schensted correspondence, see, e.g., [Knu73]). His construction of maximum k-antichains also allows to derive part (2) of Theorem 1 for orders arising from permutations by purely geometrical means. This construction implies a result of Greene [Gre74] stating that the shape of the Ferrers diagram F_α mentioned in Theorem 1 is just the shape of the Young tableaux of the permutation.

Part (1) of Theorem 1 for permutations can be deduced from the above results only by using a theorem of Schensted [Sch61]. In this paper we close this gap by providing a direct geometric proof for part (1), too (see Corollary 1). Our algorithm thus heavily participates in the "witchcraft operating behind the scenes" (Knuth [Knu73], p. 60) that seems to control the remarkable connections between chains in permutations and Young tableaux.

2 The skeleton

Let us assume that the x- and y-coordinates of all points in P are distinct (otherwise, a small perturbation will have no influence on the results). The *height* of P is the size of a maximum chain. The *height* of a point p in P is the size of a maximum chain in $\{q \in P \mid q \leq p\}$, i.e., in the subset of points smaller than p. A partition of P into antichains that will be used frequently is the *canonical antichain partition* $A_1, \ldots, A_\lambda$ of P (λ the height of P). It is obtained by assembling the elements of height i in P into one antichain A_i. Since for each point of height i there must be a point of height $i-1$, a maximum chain with one point from each A_i is easily extracted (algorithms constructing the canonical antichain partition and maximum chains are described in Section 4). Note that there is no partition of P in less than λ antichains, since otherwise an antichain would contain at least two elements of a maximum chain, which is impossible.

Following Viennot we define the *shadow of point* (x, y) *with light from the lower left* to be the set of all points (u, v) larger than (x, y), i.e., with $u \geq x$ and $v \geq y$. For an arbitrary set E of points, the *shadow of* E is the union of the shadows of the points of E, i.e., the set of all points larger than at least one point of E. The *coshadow of* E is the complement of the shadow of E. The *jump line of an antichain* A is the topological boundary of the shadow of A. Note that the points on the jump line of A belong to the shadow of A, but not to the coshadow. Similar definitions can be given for the *shadow* and the *jump line of a chain*. Here we use the *shadow of a point* (x, y) *with light from the lower right*, i.e., the set of all points (u, v) with $u \leq x$ and $v \geq y$.

Let us call the antichains of the canonical antichain partition together with their jump lines the *layers* of P. Note that the layers of P are disjoint. Equivalently, if

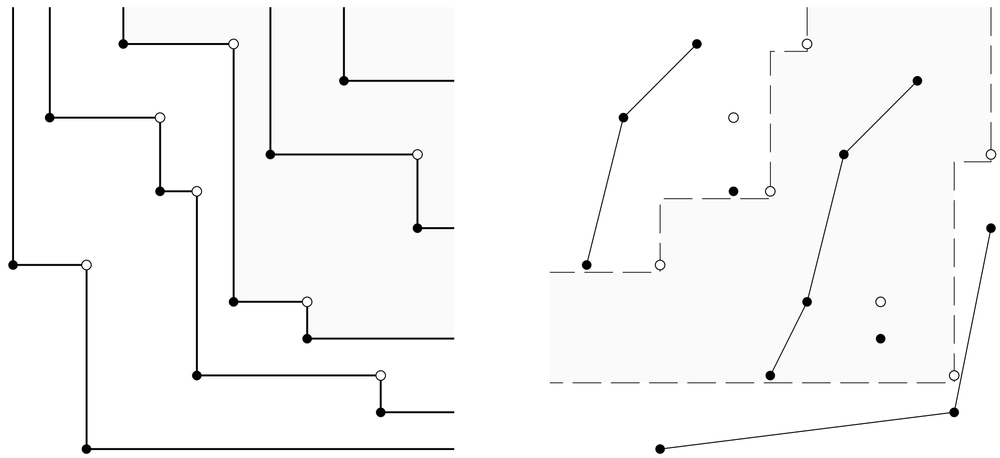

Figure 1: The skeleton of 12 points and the three regions defined by a maximum 2-chain of the skeleton each containing one chain.

$A_1, \ldots, A_\lambda$ is the canonical antichain partition of P, then all points of $\bigcup_{j \geq i} A_j$ are contained in the shadow of A_i, for $1 \leq i \leq \lambda$ (see the shadow of A_3 in the left picture of Fig. 1).

Between any two consecutive points $p = (x, y)$ and $q = (x', y')$ with $x < x'$ in the same antichain (hence, $y > y'$) there is a further vertex (x', y) on the jump line, the *skeleton point of* q (a *skeleton point of* A). Hence, an antichain A has exactly $|A| - 1$ skeleton points. The set of skeleton points of the antichains of the canonical antichain partition is the *skeleton* $S(P)$ *of* P. Obviously, $|S(P)| = |P| - \lambda$. See the left picture of Figure 1 for an example of a set P of 12 points in the plane corresponding to the permutation $(6, 10, 1, 12, 8, 3, 5, 9, 4, 11, 2, 7)$ together with its layers, the shadow of the third layer, and the skeleton $S(P)$.

Of course, if P does, then $S(P)$ also fulfills the requirement of distinct x- and y-coordinates for each point and we can again form the skeleton of $S(P)$. We set $S_0(P) = P$ and define $S_k(P) = S(S_{k-1}(P))$ to be the *k-th skeleton* of P.

The *canonical chain partition* of a set of points P is the unique minimum chain partition $C_1, \ldots, C_k$ with all points of $\bigcup_{j \geq i} C_j$ in the shadow of C_i (light from lower right!), for $1 \leq i \leq k$. Suppose a subset C_S of the skeleton $S(P)$ of P is given and let $C_1, \ldots, C_k$ be the canonical chain partition of C_S. We define the *i-th region of* C_S, for $2 \leq i \leq k$, to be the intersection of the shadow of C_{i-1} with the coshadow of C_i, i.e., the region between the jump lines of C_{i-1} and C_i, containing the jump line of C_{i-1} but excluding that of C_i (see right picture of Fig. 1). The *first region* is the coshadow of C_1 and the $(k+1)$*-st region* is the shadow of C_k. These $k+1$ regions partition the whole plane.

With these definitions we are ready to give a description of the very simple

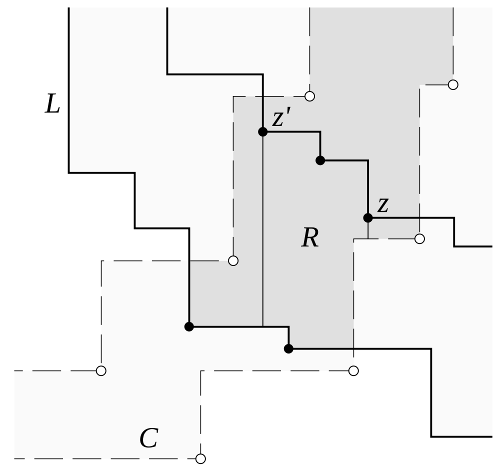

Figure 2: Jump line L crosses region R well.

iterative approach that can be used to find a maximum k-chain. Suppose that a maximum $(\ell-1)$-chain $C_{\ell-1}$ in $S_{k-\ell+1}(P)$ is given. The canonical chain partition of $C_{\ell-1}$ partitions the plane into ℓ regions $R_1, \ldots, R_\ell$. We extract a maximum chain from the points $S_{k-\ell}(P) \cap R_j$ in each region R_j, with $1 \le j \le \ell$ (this is done in the right picture of Fig. 1, where $k = \ell = 3$; i.e., a maximum 2-chain in $S(P)$ gives rise to three regions and three chains in P). In the next section we will show that the union of these ℓ chains is a maximum ℓ-chain in $S_{k-\ell}(P)$. Therefore,

(1) the iterative construction of skeletons up to $S_{k-1}(P)$,
(2) the selection of a maximum chain C_1 in the points $S_{k-1}(P)$, and
(3) the successive selection of ℓ-chains in $S_{k-\ell}(P)$ using $(\ell-1)$-chains in $S_{k-\ell+1}$ as indicated above

give a maximum k-chain C_k in P.

3 The k-chains

Suppose a subset C_S of the skeleton is given and the canonical chain partition of C_S yields $k-1$ chains that define k regions. In each region we want to retain the points of some special antichains and discard all others. We will see that the retained points in each region already admit the extraction of k suitable chains.

An antichain A of P and its jump line L *enter* a region R *vertically* (*horizontally*) if there is some point of A in R and the leftmost straight line segment of L that lies in R is vertical (horizontal). Similarly, A and L *leaves* region R *vertically* (*horizontally*) if $A \cap R$ is nonempty and the rightmost straight line segment of L in R is vertical (horizontal). We are interested in antichains and their jump lines that enter R vertically and leave it horizontally; they are said to *cross R well* (see Fig. 2).

Lemma 1 *If an antichain A crosses a region R well then any point of R that lies in the shadow of A is larger than some point of $A \cap R$.*

Proof. Let z be a point of R in the shadow of A. Denote by C the jump line of the chain bounding R from below and let L be the jump line of A. As is easily verified, every point of $L \cap R$ is larger than at least one point of $A \cap R$, since L crosses R well. Now the vertical half line below z intersects both L and C. If it intersects L first (look at z' in Fig. 2) then z is larger than a point on $L \cap R$ and hence than a point from $A \cap R$. Otherwise, z is larger than a point on C (look at z in Fig. 2) which in turn is larger than the intersection point of C and L. Hence, again z is larger than a point of $L \cap R$ and thus larger than one of $A \cap R$. □

The above lemma leads to the key property of well crossing antichains that makes them useful for our purposes.

Lemma 2 *Let R be a region of $C_S \subseteq S(P)$ and let $A_1, \ldots, A_\lambda$ be the canonical antichain partition of P. If I is the set of all indices i such that A_i is crossing R well, then the collection $\{A_i \cap R \mid i \in I\}$ is the canonical antichain partition of the underlying point set $\bigcup_{i \in I} A_i \cap R$.*

Proof. Let i_0 be the smallest index in I. Hence, all antichains A_i, with $i \in I$, lie in the shadow of A_{i_0}. A_{i_0} crosses R well; thus, by Lemma 1 (see also Fig. 2), for every point of $A_i \cap R$ $(i \in I)$ there is a smaller one in $A_{i_0} \cap R$. Hence, $A_{i_0} \cap R$ is the set of the minimal points (i.e., of height 1) in $\bigcup_{i \in I} A_i \cap R$ and the lemma follows with induction. □

Lemma 3 *Let $C_S \subseteq S(P)$ and let A be an antichain of the canonical partition of P. If m is the number of skeleton points on A that are in C_S then the number of regions of C_S crossed well by A is at least $m+1$.*

Proof. Let c be a point of C_S on the jump line L of A. Let p and q be the points of A to the left and below c that define it (see Fig. 3). Then it is obvious that L leaves the region containing p horizontally and enters that of q vertically. Of course, the leftmost straight line segment of the whole of L is vertical as the rightmost one is horizontal.

Note that if L leaves one region vertically the region of the next point of A to the right is entered vertically, too. Now consider the $m+1$ sections of L from left to right before, between, and after its m points in C_S (possibly $m = 0$); since in each section A enters its first region vertically and leaves its last region horizontally, there must be some region crossed well by A in between (see Fig. 3). □

Theorem 2 *Let C_S be a $(k-1)$-chain in the skeleton $S(P)$ of points P and let λ be the height of P. Taking a maximum chain of $P \cap R$ in each region R of C_S yields a k-chain of P of size at least $|C_S| + \lambda$.*

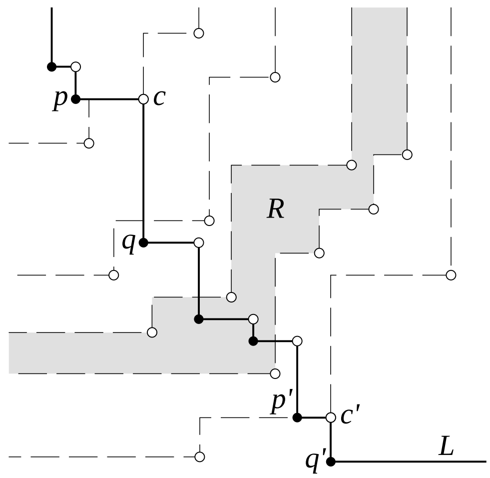

Figure 3: L crosses some region R well between c and c'.

Proof. Let $A_1, \ldots, A_\lambda$ be the canonical antichain partition of P. Each point of C_S is a skeleton point of exactly one antichain A_i. If m_i is the number of skeleton points of A_i in C_S, for $1 \leq i \leq \lambda$, then, according to Lemma 3, the antichains A_i cross the regions well in altogether at least $\sum_{1 \leq i \leq \lambda}(m_i + 1) = |C_S| + \lambda$ sections. On the other hand, by Lemma 2, each such section contributes an additional antichain to some region and hence contributes one more point to the maximum chain of that region. □

Note that Theorem 2 does not require the $(k-1)$-chain C_S to be maximum. At this point we could stop and refer to Greene's Theorem (Theorem 1) and Viennot's interpretation of F_α (see [Vie84]) to show that a k-chain found by repeated application of Theorem 2 starting with a maximum chain in the $(k-1)$-st skeleton of P is maximum. We prefer to make one step further and proof a kind of reverse to Theorem 2, thus making this paper self-contained and closing a gap left by Viennot in his skeleton approach to the Greene-Kleitman theory for permutations.

This time we consider $k-1$ regions of a k-chain C *in* P (ignoring the first and the last region) and search for maximum chains of *skeleton points* in these regions. We say that an antichain A of P and its jump line L enter a region R of C horizontally (vertically) *for the skeleton* if there is some skeleton point of A in R and the leftmost straight line segment of L in R is horizontal (vertical). The definitions for leaving vertically (horizontally) are obvious. We say that A *crosses region* R *well for the skeleton*, if L enters R *horizontally* and leaves R *vertically*. The following lemma is the counterpart of Lemma 1 and proved analogously (see Fig. 4).

Lemma 4 *If an antichain A crosses a region R well for the skeleton then any point of R that lies in the coshadow of A is smaller than some point of $S(A) \cap R$, the skeleton points of A in R.* □

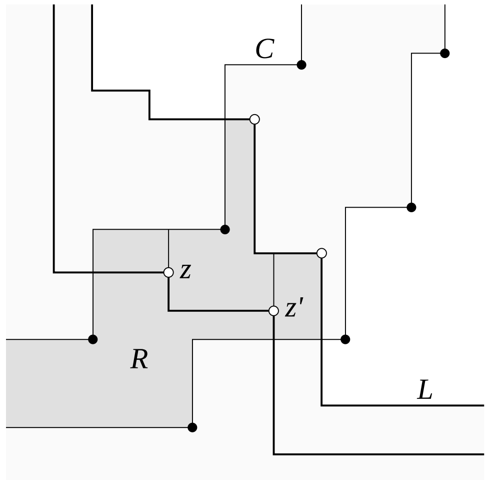

Figure 4: L crosses R well for the skeleton.

The next lemma is a comprehension of the reverses to Lemma 2, Lemma 3, and Theorem 2.

Lemma 5 *Let C be a k-chain in P. There exists a $(k-1)$-chain in the skeleton $S(P)$ with size at least $|C| - \lambda$, where λ is the height of P.*

Proof. Let $A_1, \ldots, A_\lambda$ be the canonical antichain partition of P and let R be one of the regions of C. Let I be the set of indices of antichains A_i that cross R well for the skeleton and denote the largest index in I by i_0. By Lemma 4, the skeleton points $S(A_{i_0}) \cap R$ of A_{i_0} in R are the set of maximal points among the skeleton points $\bigcup_{i \in I} S(A_i) \cap R$ in R. Hence, a partition of the skeleton points of P lying in R requires at least $|I|$ antichains and a maximal chain in $S(P) \cap R$ has size at least $|I|$.

Suppose there are m_i points of A_i in C, then we have $m_i - 1$ sections of the jump line of A_i between them (ignoring the sections before the first and after the last point). Let s_L be the leftmost and s_R the rightmost skeleton point on the jump line of A_i in such a section. It is easy to see that A_i enters the region of s_L horizontally for the skeleton and leaves the region of s_R vertically for the skeleton. Since A_i leaving a region horizontally for the skeleton implies A_i entering the region of the next skeleton point to the right horizontally for the skeleton, too, there must be a region crossed well for the skeleton in between.

Consequently, there are at least $m_i - 1$ skeleton points on the jump line of A_i each one contributing a point to the maximum chain of skeleton points in its region. In summary, this yields a $(k-1)$-chain of size at least $\sum_{1 \le i \le \lambda}(m_i - 1) = |C| - \lambda$ in the skeleton $S(P)$. □

We denote the k-chain in P obtained from a $(k-1)$-chain C_S in $S(P)$ according to Theorem 2 by $\sigma_k(C_S)$.

Theorem 3 *Let C_1 be a maximum chain of $S_{k-1}(P)$ and $C_\ell = \sigma_\ell(C_{\ell-1})$, for $2 \leq \ell \leq k$, then C_k is a maximum k-chain in P.*

Proof. By induction, suppose that C_{k-1} is a maximum $(k-1)$-chain in $S(P)$ and let λ be the height of P. If there were a k-chain $C \subseteq P$ with more points than $\sigma_k(C_{k-1})$ then C would have more than $|C_{k-1}| + \lambda$ points, according to Theorem 2. Hence, by Lemma 5, there would be a $(k-1)$-chain of size larger than $|C_{k-1}|$ in $S(P)$, a contradiction. □

Note that the proof of the above theorem also shows that the number of additional points in each application of σ_ℓ to a maximum $(\ell-1)$-chain of $S_{k-\ell+1}(P)$ is equal to the height of $S_{k-\ell}(P)$, for $2 \leq \ell \leq k$. Hence, we obtain the following corollary that is part (1) of Greene's Theorem (see Theorem 1) for permutations.

Corollary 1 *If λ_ℓ denotes the height of $S_\ell(P)$ then a maximum k-chain in P has size $\sum_{0\leq\ell\leq k-1} \lambda_\ell$.* □

4 The algorithm

To demonstrate the simplicity of the algorithm and for the sake of completeness we describe it in fairly detailed pseudocode. Certainly, to achieve the aim of being fast, the algorithm has to be somewhat more sophisticated than indicated by the sketch in Section 2. Particularly, we do not want to waste time by computing a new antichain partition for each region. Instead we make use of the theory developed in the previous section and restrict attention to the well crossing antichains in each region that will already give us the partition needed.

The first two algorithms are well known (see Fredman [Fre75], Knuth [Knu73], Viennot [Vie84]); we describe them to introduce concepts used in the the main algorithms. The first algorithm LAYERS(P) (see Fig. 5) computes the canonical antichain partition and the skeleton of P with the help of a sweep line l moving from left to right and halting at every point $p \in P$. It stores the rightmost points on the layers of the subset P' of P seen so far. For convenience, we add a dummy point $(0, +\infty)$ to P. Further, we assume a procedure *insert*(q, l) (initialized with the dummy point) that inserts a new point $q \in P$ between the two points in l with y-coordinate smaller and larger, and returns the latter one, call it p. If p is not the dummy point, it has to be removed from l by *remove*(p, l). Algorithm LAYERS not only returns the skeleton of P and the collection of layers but also computes some properties of the layers and the points of P used in subsequent algorithms. Note that by interchanging "left" with "right" in algorithm LAYERS, it computes the layer structure of P with light from the lower right, i.e., the canonical chain partition; let this algorithm then be LAYERS_lower_right(P).

The second algorithm MAXCHAIN (see Fig. 5) finds a maximum chain C of P given the canonical antichain partition $\{A_1, \ldots, A_\lambda\}$. For $q \in A_i$ there always exists a smaller point p on layer A_{i-1}. Otherwise, point q would not have been inserted

```
LAYERS(P)
insert((0, +∞), l); λ := 0;
for each q ∈ P from left to right do
      p := insert(q, l);
      if y(p) = +∞ then                    {new layer}
            λ := λ + 1;
            L := L_λ; leftmost(L) := q;
            s := (0, +∞);
      else                                  {old layer}
            remove(p, l);
            L := layer(p); L := L ∪ {q};
            s := (x(q), y(p)); S := S ∪ {s};
      layer(q) := L; nextleft(q) := p; skeleton(q) := s;
      rightmost(L) := q;
return (S, {L_1, ..., L_λ}).
```

```
MAXCHAIN({A_1, ..., A_λ})
l := +∞;
for i := λ downto 1 do
      p := rightmost(A_i);
      while x(p) > l do p := nextleft(p);
      C := C ∪ {p}; l := x(p);
return C.
```

Figure 5: Algorithms LAYERS and MAXCHAIN

into A_i but in some layer A_j with $j < i$ by algorithm LAYER. Hence, Algorithm MAXCHAIN returns a chain with points from each layer, which thus is maximum.

Procedure LAYERS needs time $O(|P| \log |P|)$ (using some balanced search tree for procedure *insert*); whereas MAXCHAIN runs in time $O(|A_1 \cup \ldots \cup A_\lambda|)$. Both use $O(|P|)$ space. Fredman [Fre75] shows that to compute the maximum chain needs $\Omega(|P| \log |P|)$ comparisons.

Suppose we have an algorithm MULTICHAIN$(P, C_S, S, \mathcal{A})$ that takes a point set P, a $(k-1)$-chain C_S of the skeleton S of P, and the canonical antichain partition $\mathcal{A}$ of P to compute a k-chain of P according to Theorem 2. Then Algorithm MAXMULTICHAIN(k, P) (see Fig. 6), operating along the lines of Theorem 3, returns a maximum k-chain of P.

Now let us consider the main algorithm MULTICHAIN$(P, C_S, S, \mathcal{A})$ (see Fig. 6) in more detail. In a first step the minimum chain partition $\mathcal{C}_S$ of C_S in $k-1$ chains $C_1, \ldots, C_{k-1}$ is constructed. Then we have to find that fragments of the antichains in $\mathcal{A}$ where they cross regions of C_S well (see Theorem 2). This is done with the help of a sweep line l which goes from left to right and halts at every point q of P.

```
MAXMULTICHAIN(k, P)
(S, 𝒜) := LAYERS(P);
if k = 1 then return MAXCHAIN(𝒜);
else
    C_S := MAXMULTICHAIN(k − 1, S);
    return MULTICHAIN(P, C_S, S, 𝒜).
```

```
MULTICHAIN(P, C_S, S, 𝒜)
(D, 𝒞_S) := LAYERS_lower_right(C_S)          {𝒞_S = {C_1, ..., C_{k−1}}};
for i := 1 to k − 1 do l(i) := y(leftmost(C_i));
l(0) := −∞; l(k) := +∞;
region((0, +∞)) := 0;
for each q ∈ P from left to right do
    find r with l(r − 1) ≤ y(q) < l(r);
    region(q) := r; A := layer(q, 𝒜);
    p := nextleft(q, 𝒜); s := skeleton(q);
    if region(p) = r then                     {left neighbor p same region as q}
        if collect(A) = true then
            add q to fragment(A);
    else
        if s ∈ C_S then                       {skeleton point between p and q is in C_S}
            i := index of layer(s, 𝒞_S);
            l(i) := y(nextright(s, 𝒞_S));
        if y(s) < l(r) or s ∈ C_S then        {jump line leaves region of p horizontally}
            if collect(A) = true then
                add fragment(A) to antichains(region(p));
                collect(A) := false;
        if y(s) ≥ l(r) then                   {jump line enters r vertically}
            collect(A) := true;
            fragment(A) := {q};
for each A ∈ 𝒜 do
    if collect(A) = true then
        add fragment(A) to antichains(region(rightmost(A)));
for r := 1 to k do
    C_P := C_P ∪ MAXCHAIN(antichains(r));
return C_P;
```

Figure 6: Algorithms MAXMULTICHAIN and MULTICHAIN

To determine the region of q, l holds the y-coordinates of its intersections with the jump lines of the chains C_i. After initialization to the y-coordinate of the leftmost point of C_i, for $1 \leq i \leq k-1$, $l(i)$ is adjusted when l passes some skeleton point $s \in C_i$, i.e., when l halts at $q \in P$ with skeleton point s.

Now if l halts at $q \in P$, at first its region is determined, suppose it is the r-th region R. Then, we have to check the left neighbor p on the antichain A of q. If p is in the same region R as q and the jump line entered R vertically (i.e., the flag *collect*(A) is true), we add q to the present collection of a possibly useful fragment of antichain A in region R. If p lies in another region then the skeleton point s between p and q perhaps lies in C_S and the boundary value on l has to be adjusted as noted above.

If the skeleton point s lies in R (i.e., $y(s) < l(r)$) or is in C_S then certainly the jump line leaves the region of p horizontally and a collection of a fragment of A ending with p can be finished and be added to the antichains of region R. If s lies outside R (i.e., $y(s) \geq l(r)$) then the jump line enters region R vertically and a new collection of a possibly useful fragment of A is started. Note that the jump line may leave the region of p horizontally and nevertheless s is outside R and not in C_S. Though the fragment up to p is useful it is abandoned and a new collection started with p. But this is of no harm since we only need one useful fragment between two points s_L and s_R of C_S on the jump line (see Lemma 3) and as is easily seen the rightmost useful fragment between s_L and s_R will not be abandoned (of course, if C_S is maximum, there is exactly one useful fragment between s_L and s_R according to Theorem 3).

Finally, for all antichains of P, the collection of their rightmost fragments is finished and assigned to the corresponding region. The maximum chains on the antichain fragments of each region are computed and their union returned as the k-chain searched for.

The careful reader may ask what happens if S becomes empty during recursion in MAXMULTICHAIN (i.e., $\mathcal{A} = \{\{p\} \mid p \in P\}$) and nevertheless $k > 1$. But then MULTICHAIN$(P, \emptyset, \emptyset, \mathcal{A})$ returns just P, as is easily checked. Hence, MAXMULTICHAIN(k, P) always computes a maximum k-chain (i.e., of maximal size among all unions of k chains) that may perhaps be covered by fewer than k chains.

Theorem 4 *Algorithm* MULTICHAIN$(P, C_S, S, \mathcal{A})$ *takes time* $O(|P| \log |P|)$ *and space* $O(|P|)$. *Consequently, Algorithm* MAXMULTICHAIN(k, P) *needs time* $O(k|P| \log |P|)$ *and space* $O(k|P|)$ *to compute a maximum* k*-chain in* P.

Proof. Since the skeleton $S(P)$ has size always smaller than P, $|C_S| \leq |P|$ and the computation of the layer structure of C_S takes time $O(|P| \log |P|)$. In Algorithm MULTICHAIN the sweep line l halts at every point of P. The only time consuming step in the processing of such a point is to locate it in the k regions on l, which can be done in time $O(\log k)$, hence in time $O(\log |P|)$. Therefore, a sweep of line l needs at most time $O(|P| \log |P|)$. Algorithm MAXCHAIN$(\mathcal{L}_r)$, where $1 \leq r \leq k$ and $\mathcal{L}_r$ is a collection of antichains in the r-th region, takes time $O(|\bigcup \mathcal{L}_r|)$. But $\sum_{1 \leq r \leq k} |\bigcup \mathcal{L}_r| \leq |\bigcup \mathcal{A}| = |P|$, and so the last loop in MULTICHAIN is done in no more than $O(|P|)$ steps. It is easily checked that the space of all data structures needed is linear in $|P|$.

The k recursions in MAXMULTICHAIN(k, P) take time $O(|P| \log |P|)$ each. Together, this gives the time bounds of the theorem. Again, each recursion needs space linear in $|P|$. □

5 Conclusion

We have presented an $O(kn \log n)$ time algorithm for the maximum k-chain problem on 2-dimensional orders. Note that after sorting the points we may assume them to lie on a $n \times n$ grid. We then can use the data structures of van Emde Boas [vEB77] that support insertion, deletion, predecessor and successor finding in a subset of $\{1, \dots, n\}$ in $O(\log \log n)$ time and $O(n)$ space on a RAM in the unit cost model. Hence, the algorithm can be made to work in time $O(n \log n + kn \log \log n)$.

By flipping $(x, y) \to (-x, y)$ of the point set P we easily obtain an algorithm for maximum k-antichains. It would be interesting to know, whether the structural and algorithmical results of this paper depend on this symmetry between chains and antichains in dimension 2 essentially. This question can be separated into two problems:

(1) Is there a generalization of skeletons to higher dimensions, which provides a description of the size of a maximum k-chain (k-antichain) of P in terms of the size of a maximum k-chain (k-antichain) of $S(P)$?

(2) What is the complexity of the maximum k-chain (k-antichain) problem in fixed dimension $d \geq 3$?

References

[Fre75] M. L. Fredman. On computing the length of longest increasing subsequences. *Discr. Math.*, 11:29–35, 1975.

[GK76] C. Greene and D. J. Kleitman. The structure of sperner k-families. *J. Comb. Th. (A)*, 20:41–68, 1976.

[Gre74] C. Greene. An extension of Schensted's theorem. *Adv. in Math.*, 14:254–265, 1974.

[Gre76] C. Greene. Some partitions associated with a partially ordered set. *J. Comb. Th. (A)*, 20:69–79, 1976.

[Knu73] D. E. Knuth. *The Art of Computer Programming III. Sorting and Searching*, volume 3. Addison-Wesley, 1973.

[SL92] M. Sarrafzadeh and R. D. Lou. An optimal algorithm for the maximum 3-chain problem, to appear in SIAM J. on Comp., 1992.

[SLL90] M. Sarrafzadeh, R. D. Lou, and D. T. Lee. An optimal algorithm for the maximum two-chain problem. In *Proc. of the 1. Ann. ACM-SIAM Symp. on Discr. Alg.*, pages 149–158, 1990.

[MW92] J. Matoušek and E. Welzl. Good splitters for counting points in triangles. *J. of Algorithms*, 13:307–319, 1992.

[Sch61] C. Schensted. Longest increasing and decreasing sequences. *Canad. J. Math.*, 13:179–191, 1961.

[vEB77] P. van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Inf. Proc. Letters*, 6:80–82, 1977.

[Vie77] G. Viennot. Une forme géométrique de la correspondance de Robinson-Schenstedt. In D. Foata, editor, *Combinatoire et Représentation du Groupe Symétrique*, pages 29–58. Lecture Notes in Mathematics 579, Berlin Heidelberg New-York, 1977.

[Vie84] G. Viennot. Chain and antichain families, grids and young tableaux. In *Orders: Description and Roles*, pages 409–463. North-Holland Math. Stud. 99, Amsterdam-New York, 1984.