SERIE B — INFORMATIK

Trapezoid Graphs and Generalizations, Geometry and Algorithms[◇]

Stefan Felsner*
Rudolf Müller**
Lorenz Wernisch***

B 94–02
January 1994

Abstract

Trapezoid graphs are a class of cocomparability graphs containing interval graphs and permutation graphs as subclasses. They were introduced by Dagan, Golumbic and Pinter [DGP]. They propose an $O(n^2)$ algorithm for chromatic number and a less efficient algorithm for maximum clique on trapezoid graphs. Based on a geometric representation of trapezoid graphs by boxes in the plane we design optimal, i.e., $O(n \log n)$, algorithms for chromatic number, weighted independent set, clique cover and maximum weighted clique on such graphs. We also propose generalizations of trapezoid graphs called k-trapezoidal graphs. The ideas behind the clique cover and weighted independent set algorithms for trapezoid graphs carry over to higher dimensions. This leads to $O(n \log^{k-1} n)$ algorithms for k-trapezoidal graphs. We also propose a new class of graphs called *circle trapezoid graphs*. This class contains trapezoid graphs, circle graphs and circular-arc graphs as subclasses. We show that clique and independent set problems for circle trapezoid graphs are still polynomially solvable. The algorithms solving these two problems require algorithms for trapezoid graphs as subroutines.

Mathematics Subject Classification (1991). 06A07, 05C85, 68R10.

Key words. Algorithms, partially ordered sets, order dimension, trapezoid graphs.

[◇]The research of the first author was supported by DFG under grant FE-340/2-1. The research of the third author was supported by the Basic Research Action of the European Community under contract No. 7141 (project ALCOM II).

*Bell Communications Research, 445 South Street, Morristown, NJ 07962, U.S.A., and
Institut für Informatik, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, Germany, E-mail: FELSNER@INF.FU-BERLIN.DE.

**Technische Universität Berlin, Fachbereich Mathematik, Str. des 17. Juni 135, D-10623 Berlin, Germany, E-mail: MUELLER@MATH.TU-BERLIN.DE.

***Institut für Informatik, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, Germany, E-mail: WERNISCH@INF.FU-BERLIN.DE.

1 Introduction

Trapezoid graphs were introduced by Dagan, Golumbic, and Pinter [DGP]. Consider a channel, i.e., a pair of two horizontal lines. A *trapezoid* between these lines is defined by two points on the top and two points on the bottom line. A graph is a *trapezoid graph* if there exists a set of trapezoids corresponding to the vertices of the graph such that two vertices are joined by an edge iff the corresponding trapezoids intersect (see Figure 1). Dagan, Golumbic, and Pinter propose an algorithm computing the minimum number of colors in a proper coloring of such a graph in time $O(n^2)$ and less efficient backtracking algorithm finding a maximum clique in such graph (throughout the paper we assume that n is the number of vertices of the graph or order in question).

The problem of finding maximal cliques or minimal colorings for trapezoid graphs arises in connection with channel routing problems in VLSI design. Given some labeled terminals on the upper and lower side of a two sided channel, terminals with the same label will be connected in a common net. Such a net may be modeled by a trapezoid connecting the rightmost resp. leftmost terminals with the same label. Nets then may be routed without intersection iff the corresponding trapezoids do not intersect, i.e., iff they are independent. The number of colors needed to color the trapezoid graph is the number of layers needed to rout the nets without intersection.

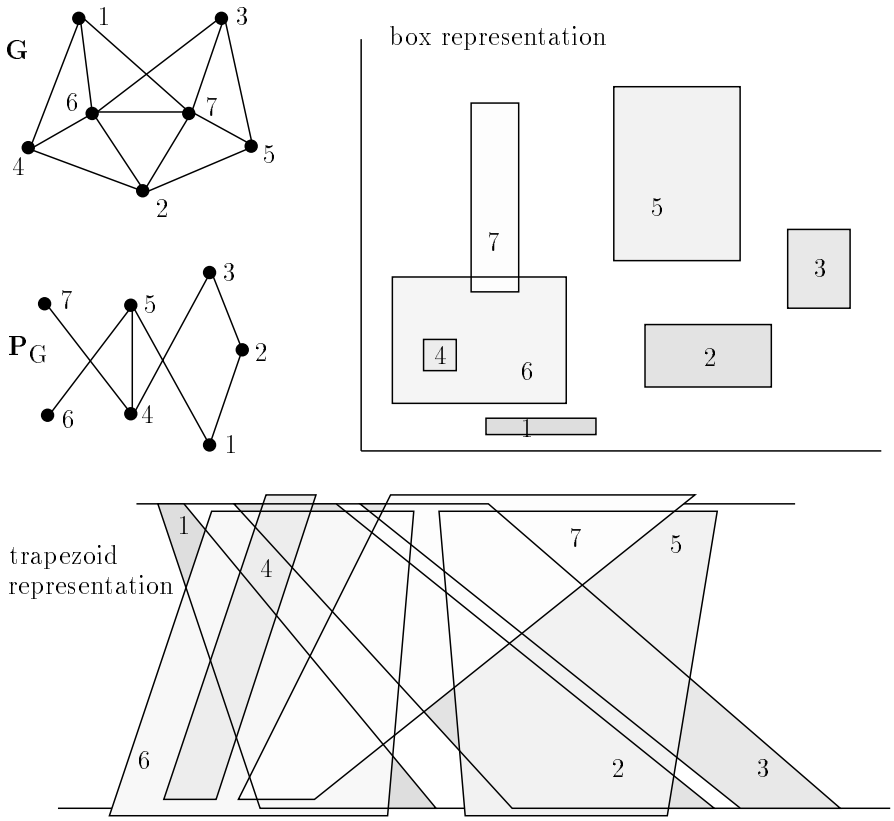

Figure 1. A trapezoid graph **G**, the order **P** and two representations.

For our algorithms we will make use of another equivalent characterization of trapezoid graphs. To give this alternative characterization it is convenient to fix some terminology. If $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$ are points in $\mathbb{R}^k$, then x is said to be *dominated* by y, denoted $x < y$, if x_i is less than y_i for all $i = 1, \ldots, k$. The order thus given between points in $\mathbb{R}^k$ is also called *dominance order*. This order can be extended to *boxes*, i.e., sets of the form $\{(x_1, \ldots, x_k) \in \mathbb{R}^k : l_i \leq x_i \leq u_i,\ 1 \leq i \leq k\}$ where $(l_1, \ldots, l_k)$ is the *lower corner* and $(u_1, \ldots, u_k)$ is the *upper corner* of the box. A box b dominates a box b' if the lower corner of b dominates the upper corner of b'. Note that points may be understood as boxes where the lower and upper corner coincides. If one of the two boxes dominates the other we say that they are *comparable*. Otherwise they are *incomparable*. Now the vertices of trapezoid graph may be represented by boxes with two boxes incomparable iff the corresponding vertices are joined by an edge.

The connection between the box representation and the trapezoid representation of a trapezoid graph is the following. Interpret the points on the lower of the two lines of the channel as lying on the x-axis and that of the upper line as lying on the y-axis of the euclidean plane. Each trapezoid then corresponds to an axis-parallel box in the plane whose projection on the x- and y-axis coincides with the lower and upper side of the trapezoid (see Figure 1). It is easily seen that two trapezoids are disjoint exactly if the corresponding boxes are comparable.

What makes the box representation useful is the additional dominance order on boxes that may be exploited by sweep line algorithms. All computation is done in a single sweep leading to $O(n \log n)$ algorithms for clique, independent set and cover problems on trapezoid graphs. Hence, these graphs are another class of graphs where very efficient algorithms for such problems can be given. There exists a lower bound for the number of comparisons needed to compute maximum increasing subsequences in permutations, Fredman [Fre]. Permutations correspond to permutation graphs in such a way that increasing sequences correspond to either cliques or independent sets. As permutation graphs are trapezoid graphs Fredman's bound shows that our algorithms are optimal in the same sense.

Algorithms for trapezoid graphs should be compared with algorithms for general cocomparability graphs. For these graphs the maximum independent set and the minimum clique cover problem can be solved in $O(n^2 \log n)$, see [MCSp]. The bottleneck of the computation is the complexity of transitive orientation. The maximum clique and chromatic number problems on cocomparability graphs seem to be harder. To the best of our knowledge the complexity is dominated by finding a maximum matching in a bipartite graph. The time needed to solve this problem is almost $O(n^{2.5})$ (see [ABMP]), and $O(n^3)$ in the weighted case (see [PaSt]).

In Section 2 we give some definitions and replace graph terminology by order terminology that proves to be more convenient in designing our algorithms. We assume the vertices of the trapezoid graph to have some weights. To compute maximum weighted cliques or independent sets turns out to impose no additional difficulty. In Section 3 we present an algorithm computing maximum weighted independent set and a minimum clique cover at the same time (or in order terminology, a maximum

weighted chain and a minimum antichain partition). We also show how to extend this algorithm from boxes in the plane to boxes in $\mathbb{R}^k$. Section 4 shows how to compute a minimum coloring (or a minimum chain partition). Unfortunately, this algorithm cannot be turned into an efficient one finding a maximum weighted clique (maximum weighted antichain). Hence, a different approach is proposed in Section 5 giving an efficient algorithm for the last problem.

In Section 6 we discuss a new class of graphs, called circle trapezoid graphs. A *circle trapezoid* is the region between two non-crossing chords of a circle. Alternatively, it is the convex hull of two disjoint arcs on the circle. *Circle trapezoid graphs*, *CT-graphs* for short, are the intersection graphs of families of circle trapezoids on a fixed circle. It is easily seen, that CT-graphs are a common generalization of trapezoid graphs, circle graphs and circular-arc graphs. We show, that in this large class of graphs the maximum clique and maximum independent set problems can still be solved in polynomial time.

2 Trapezoid graphs and trapezoid orders

The *k-dimensional box representation* (V, l, u) of a graph $\mathbf{G} = (V, E)$ consists of mappings $l: V \to \mathbb{R}^k$ and $u: V \to \mathbb{R}^k$ such that $l(v)$ is the lower and $u(v)$ the upper corner of a box $\mathrm{box}(v)$ where two vertices of the graph are joined by an edge iff their corresponding boxes are incomparable. If a graph has a such a representation it is a *k-trapezoid graph*. If we additionally have a *weight* $w: V \to \mathbb{R}$ on the vertices of $\mathbf{G}$ then the k-trapezoid graph is *weighted*. The weight of a clique, i.e., a set of mutually joined vertices in the graph, is the sum of the weights of its elements. Similarly, the weight of an independent set, i.e., a set of vertices with no two of them joined by an edge, is the sum of the weights of its elements. We are mainly interested in the case $k = 2$ where we simply deal with *trapezoid graphs*.

As already mentioned in Section 1, we switch to the richer structure given by the dominance order on the boxes of a box representation. Let the boxes of a box representation of a trapezoid graph together with the dominance order be the corresponding *trapezoid order*. A set of mutually comparable elements of an order is a *chain* as a set of mutually incomparable elements is an *antichain*. Recall that two boxes are incomparable iff the corresponding vertices of the trapezoid graph are joined. Let $\mathbf{G}$ be a trapezoid graph and $\mathbf{P}$ be a corresponding trapezoid order. Then it is easily verified that

- A minimum clique cover of $\mathbf{G}$ is a minimum antichain partition of $\mathbf{P}$.
- A maximum weighted independent set in $\mathbf{G}$ is a maximum weighted chain in $\mathbf{P}$.
- A minimum coloring of $\mathbf{G}$ is a minimum chain partition of $\mathbf{P}$.
- A maximum weighted clique in $\mathbf{G}$ is a maximum weighted antichain in $\mathbf{P}$.

A *maximal element* of a dominance order is one with no element dominating it. Each chain has exactly one maximal element. In contrast to the weight $w(v)$ of a

box v in a trapezoid order we will often attribute a *chain weight* $W(v)$ to v which is the maximum weight of a chain with v as its maximal element.

Note that in the limiting case the box representation (V, l, u) of a trapezoid graph (V, E) may consist of points, i.e., $l(v) = u(v)$, for all $v \in V$. Such graphs are known as *permutation graphs* and the points with the dominance order in the plane as *2-dimensional order* (see, e.g., [Gol]). We denote such an order by (V, p) with $p(v) = l(v) = u(v)$. Before giving the actual algorithms for the trapezoid orders we will sometimes recall algorithms for 2-dimensional orders since they are easier to grasp while showing important features extendible to the general case.

We will often have to maintain a finite set of real numbers such that values may be inserted or deleted from it and the predecessor or successor of a given query value can be found. Using balanced trees (e.g., red-black trees described in Cormen, Leiserson, Rivest [CLR]) all these operations can be done in $O(\log n)$ time and linear space. If we further assume the benefits of a random access machine and assume that the values are taken from a finite range U then the above operations take only $O(\log \log n)$ time and linear space when implemented on a data structure of van Emde Boas [vEB]. Hence, under these assumptions, the $\log n$ factor in the running time of the algorithms for 2-dimensional trapezoid orders may be replaced by a $\log \log n$ factor.

Throughout the paper we assume that the points $l(v)$ and $u(v)$ of a box representation have mutually different x- and y-coordinates. Otherwise, we may obtain a box representation of the same order fulfilling this requirement by perturbing the corner points with two line sweeps in the following way. Points with the same x-coordinates are perturbed slightly such that points which are lower corners have smaller x-coordinates than such which are upper corners. A similar perturbation is done for the y-coordinate. The x- and y-coordinate of a point $p \in \mathbb{R}^2$ will be denoted by p_x and p_y, resp. We will always use a vertical sweep line L going from right to left, i.e., from lower to higher x-coordinates.

3 Minimum antichain partition and maximum chain for k-trapezoidal orders

We first give a brief description of an algorithm solving the maximum chain problem for a 2-dimensional order (V, p) in the weighted case. Let the weights be given by $w: V \to \mathbb{R}^+$. First, the points are sorted, so that we can access them by increasing x-coordinate, i.e., from left to right. Secondly, we compute a function $W: V \to \mathbb{R}$, where $W(v)$ is the chain weight of v, i.e., the weight of a maximum weighted chain having v as its maximal element.

$W(v)$ is computed with the aid of a sweep line L moving from left to right and halting at every point $p(v)$. We maintain a set M of weighted markers on L so that the weight $W(m)$ for $m \in M$ is just the weight of a maximum weighted chain on the set of points dominated by m, i.e., on $\{v \in V : p(v) < m\}$. For each $m \in M$ origin(m) is the maximal element of the maximum weighted chain

dominated by m. When reaching a point $p(v)$ we find the first marker m below $p(v)$ on L, set $W(v) = W(m) + w(v)$ and establish a link from v to origin(m). To update L we position a new marker m' with $W(m') = W(v)$ and origin$(m') = v$ at the y-coordinate of $p(v)$. Then we remove those markers above m' that have smaller weight. Note that also the number of markers removed in one step may be large, the overall number of insertions and removals of markers on L cannot exceed $2n$. Finally, starting from a point v with maximum chain weight $W(v)$ we backtrack along the links to construct a heaviest chain.

Now we mimic this algorithm for the case where the box representation (V, l, u) of a trapezoid order $\mathbf{P}$ is given. Essentially, the idea is to separate the action taken by the algorithms for 2-dimensional orders whenever the sweep line reaches a new element into two parts. The first part of the action, located at $l(v)$, is to compute the chain weight W of the new element v. This is done by finding the element v' of maximum chain weight among the elements with $u(v') < l(v)$ and link v to v'. Note that the maximum weight of v' implies that v' was the maximum element of its chain. The second part of the action, located at $u(v)$, is to make the chain weight of v available for further elements. The main difference to the permutation graph algorithm is that before inserting the information corresponding to v into the structure M we have to check whether the information is still relevant when released. The reason is that there might be an element v' with $W(v') > W(v)$ whose box is completely dominated by the upper corner of v's box. Again, the weight of marker $m \in M$ will be equal to the weight of a maximum weighted chain on the boxes dominated by m, i.e., on the elements $v \in V$ with $u(v) < m$, in particular the weights on M are increasing with increasing y-coordinate.

The algorithm for computing a maximum weighted chain in a box representation is given next. For convenience, we initialize the sweep line with a dummy point d with $W(d) = 0$ and origin$(d) =$ nil, such that d is below all points that will ever be inserted into L.

```
MAXCHAIN(V, l, u, w)
for each p from left to right do
    m ← first marker below p on L
    if p = l(v) for some v ∈ V then
        W(v) ← W(m) + w(v)
        link(v) ← origin(m)
    if p = u(v) for some v ∈ V then
        if W(v) > W(m) then
            insert a new m_v at p_y in L
            W(m_v) ← W(v)
            origin(m_v) ← v
            remove all m' that are higher and lighter than m_v from L
v ← origin(uppermost(L))
C ← {v}
while link(v) ≠ nil do
    v ← link(v)
    C ← C ∪ {v}
return C
```

Lemma 1 *At the end of the main loop in MAXCHAIN the following invariant holds true. If y is an arbitrary point on L and m the next marker below y, then a maximum weighted chain dominated by point y has weight $W(m)$.*

Proof: If L has stopped at some point $p = l(v)$ no new box has become available to increase any maximum weighted chain and no weight of any marker has been changed. But note that the weight of a maximum chain with maximal element v has weight $W(m)+w(v)$, for m the marker below $l(v)$, by the invariance assumption.

On the other hand, suppose L has stopped at $p = u(v)$. If $y < u_y(v)$ or if no new marker is inserted in the sweep line, there can neither be a new maximum weighted chain nor a new marker below y. Hence we assume $y \geq u_y(v)$ and that a new marker m_v has been inserted at height $u_y(v)$, i.e., there is a new chain with weight $W(v) = W(m_v)$ available for points above $u_y(v)$. Let m' and m be the marker immediately below y before and after the insertion of m_v. If $m' = m$ then $W(m') > W(m_v)$ (otherwise, m' would have been removed) and $W(m) = W(m')$ remains optimal among all chain weights. If $m' \neq m$ then m can only be m_v and $W(m_v) > W(m')$ either by the condition for removing markers or by the condition for the insertion of m_v. Since $W(m')$ was optimal among all chain weights save the new one ending in v, $W(m_v)$ surely is an optimal weight for y now. □

Of course, Lemma 1 implies that Algorithm MAXCHAIN computes a maximum weighted chain, since all boxes are dominated by the uppermost point on L after the sweep has completed.

As already noted the sweep line can be implemented so that find, insert and delete operations require $O(\log n)$ time. It is easily seen, that $3n$ is an upper bound for the number of these operations. This proves a $O(n \log n)$ time bound.

The unweighted case can be simulated by unit weights. As the weights of all markers are different the number of markers on L in the unweighted case cannot exceed the length of a maximum chain in $\mathbf{P}$. If ω is the size of a longest chain in $\mathbf{P}$ then all steps can be carried out in $O(n \log \omega)$. If each element of $\mathbf{P}$ has unit weight, then no two elements with the same chain weight are comparable. Hence, collecting the elements of chain weight i in a set A_i yields a partition $A_1, \ldots, A_\omega$ of $\mathbf{P}$ into antichains. It is easily seen that the maximum weighted chain must contain one element of A_i, $i = 1, \ldots, \omega$. This proves this antichain partition to be minimal since a partition into fewer antichains would force at least two elements of the chain into one antichain, which is impossible. Hence, a minimal antichain partition is a byproduct of algorithm MAXCHAIN. We summarize these remarks in

Theorem 2 *A maximum weighted chain and a minimum antichain partition of a trapezoid order on n points, given its box representation, can be computed in $O(n \log n)$ time and linear space.* □

Now assume, that a box representation of $\mathbf{P}$ in some higher dimension $k > 2$ is given. We use dynamic range trees (see, e.g., Smid [Smi]) for the construction of a maximum chain for a point set in k dimensions. We need a data structure for a point set P in $k-1$ dimensional space that, for a given query point q, allows to find some $p \in P$ with maximum chain weight $W(p)$ among all points of P that are dominated by q. We also want to insert new points with some given chain weight. Given such a data structure, it is easy to compute a maximum weighted chain for a point set P in k dimensions. A sweep line visits all points ordered by increasing last coordinate. For each point q on the sweep line a point p is found in the range tree that has maximum chain weight $W(p)$ among all points dominated by q in the first $k-1$ coordinates. But since all points with smaller values than q in their last coordinate have been swept and have already been inserted in the range tree, p also has maximum weight among points dominated by q in all k dimensions. Hence, we may insert q in the range tree with chain weight $W(q) = W(p) + w(q)$. Along with $W(q)$ we may also store a link to point p. After insertion of all points a maximum weighted chain is easily found. At first, a point p_m with the highest weight ever computed during the sweep is searched. Then, beginning with p_m, the chain is extracted by following the corresponding chain of links. Again, if all points have unit weight, $W(p)$ is the height of p and hence a minimum antichain partition is found as byproduct.

With the following changes the above approach is easily adapted to compute a maximum weighted chain for a trapezoid order with a box representation. If point q on the sweep line corresponds to a lower point $l(v)$ of some box v we calculate $W(v) = W(p) + w(v)$ as above but do not yet insert q in the tree. If $q = u(v)$ for some box v we insert q in the range tree with chain weight $W(q) = W(v)$ that has already been calculated before.

For convenience, let us briefly recall how such a $d = k-1$ dimensional range tree T works. Let the d coordinates be denoted by $x_1, \ldots, x_d$. Points P are represented

by the leaves of a binary tree T ordered according to their x_d-coordinate. If $d = 1$ each node t of T points to the leaf of some point p with chain weight $W(p)$ maximal among all weights in the leaves below t, i.e., leaves of the subtree rooted at t. If $d > 1$ node t points to a $d-1$ dimensional range tree with respect to the first $d-1$ coordinates for the points in the subtree of t. For a query point q the point of P with smallest x_d-coordinate greater than that of q is searched. Let the search path be S_q. Let L_q be the set of left children of nodes in S_q that are not itself in S_q. It is easily seen that each point with x_d-coordinate smaller than or equal to that of q has a leaf below some node in L_q. Hence, to find a point p with $W(p)$ maximal among all points dominated by q we proceed as follows. If $d = 1$ we check all leaves pointed to by the nodes in L_q and return the leaf with maximum weight. If $d > 2$ the range trees in nodes $t \in L_q$ allow to find points p_t with maximum weight among leaves below t and dominated by q in the first $d-1$ coordinates. In this case the point searched for is that with maximum weight among points p_t, for $t \in L_q$. On the other hand, if we want to insert point q into tree T this may be done by first inserting it in the main tree and then inserting it in all secondary range trees at nodes along the insertion path, if $d > 1$. If $d = 1$, pointers along the insertion path are set to the leaf belonging to q if its weight is the new maximum in the corresponding subtree.

It is easily seen that a query takes time $O(\log^d n)$ if all trees are balanced. If some trees become unbalanced during an insertion they must be rebalanced and it can be shown that this takes amortized time $O(\log^d n)$. For $d = 1$ we need linear space. And since a point is contained in at most $\log n$ secondary trees if $d > 1$, the total amount of space is $O(n \log^{d-1} n)$, by induction. We leave it to the reader to supplement the omitted details that give a complete proof of the following statement.

Theorem 3 *If an order* $\mathbf{P} = (V, P)$ *is given by a box representation in* $\mathbb{R}^k$*, then a minimum antichain partition and a maximum chain of* $\mathbf{P}$ *can both be obtained in* $O(n \log^{k-1} n)$ *time and* $O(n \log^{k-2} n)$ *space.* □

4 Chain partitions of trapezoid orders

In this section we show how to partition a trapezoid order $\mathbf{P}$ into chains such that the number of chains used is minimal. Of course, this only makes sense if we assume unit weights on the elements of $\mathbf{P}$. Again, we begin with a short description of a similar algorithm for 2-dimensional orders which we then adapt for the case of a given box representation of $\mathbf{P}$.

An optimal chain partition for a point set can be obtained by a sweep of a line L from left to right in the following way. Assume the set of points to the left of the current position of L to be already optimally partitioned into chains. On L the maximal elements of the chains of this partition are maintained ordered by y-coordinates. When reaching a new point p we search for the point q on L which has maximal y-coordinate among all points on L that are below p. If q exists then p is appended as new maximal element to the chain of q, otherwise, p does not dominate

any chain of the actual partition and we initialize a new chain consisting of p only. Finally, L is updated by inserting p and removing q.

Now suppose, that $\mathbf{P}$ is given by a box representation (V, l, u). We have to separate the action that has to be taken when the sweep line reaches a new element into two parts. The first part of the action, located at $l(v)$, is to find the chain of the already existing partition that will be extended by v. The second part, located at $u(v)$, is to make the chain with maximum v available for further elements. A chain C with maximum element v will be called *closed* as long as $u(v)$ has not been visited by L, otherwise C is *open*.

The algorithm for computing a minimum chain partition in a box representation is given as follows. We initialize the sweep line with a dummy point d such that d is below all points that will ever be inserted into L.

MINCHAINPARTITION(V, l, u)
for each p from left to right **do**
    $q \leftarrow$ first element below p on L
    if $p = l(v)$ for some $v \in V$ **then**
        if $q = u(w)$ for some $w \in V$ **then**
            chain$(v) \leftarrow$ chain$(w) \cup \{v\}$
            remove q from L
        else $(q = d)$
            chain$(v) \leftarrow \{v\}$
    if $p = u(v)$ for some $v \in V$ **then**
        insert p at p_y in L
return $\{$ chain$(v) : u(v) \in L\}$

The time consuming operations in this algorithm are the search, insert and remove operations for points on the sweep line L. With the use of a balanced search tree the running time of the algorithm is in $O(n \log n)$. If we assume the points to be presorted, the running time is in $O(n \log \alpha)$ where α is the number of chains in the partition.

To prove that the chain partition found by this algorithm is minimum we show how to extract an antichain from $\mathbf{P}$ that contains an element from each chain in the partition. Let $\mathcal{C} = \{C_1, \ldots, C_\alpha\}$ be the chain partition found. Let v be the last element that opened a new chain, say C_α. Note that $l_x(v)$ is larger then $l_x(v')$ if v' is the minimal element of a chain C_i with $i \neq \alpha$. Let $\mathcal{C}_1$ be the set of chains containing an element v' with $l(v') < l(v)$ and $l_x(v) < u_x(v')$. All the chains in $\mathcal{C}_1$ were closed while $l(v)$ was processed. Let U be the set consisting of such an element v' from every chain in $\mathcal{C}_1$. From the definition of U it is clear that $U \cup \{v\}$ is an antichain. Define $\mathcal{C}_2$ as the remaining set of chains, i.e., $\mathcal{C}_2 = \mathcal{C} \setminus (\mathcal{C}_1 \cup \{C_\alpha\})$. Let $X(v)$ be the set of elements $v' \in V$, such that either $l(v')$ or $u(v')$ is contained in the quarter-plane $\{(x, y) : x \leq l_x(v) \text{ and } y \geq l_y(v)\}$. It is easily seen that every chain $C \in \mathcal{C}_2$ contains elements of $X(v)$. Let C^* be the subchain of C induced by the elements in $X(v)$ and $\mathcal{C}_2^*$ be the set of these subchains. The next lemma states the

crucial property of $\mathcal{C}_2^*$.

Lemma 4 *The chain partition $\mathcal{C}_2^*$ of the order induced by $X(v)$ is exactly the chain partition generated by* MINCHAINPARTITION*, when the input consists of the boxes of elements in $X(v)$ only.*

Proof: Let L be the sweep line for input (V, l, u) and L^* be the sweep line for the restricted input, i.e, $X(v)$ replaces V. The lemma is an easy consequence of the following invariant: *Considered at the same x-coordinate, $x \leq l_x(v)$, the restriction of L to the half line above $l_y(v)$ and L^* are identical.* This is certainly true at the beginning when both lines are empty. Now suppose they are equal and L meets point p. We distinguish four cases.

First consider the situation $p = l(v')$ and $v' \notin X(v)$. Since $v' \notin X(v)$ we have $l_y(v') < l_y(v)$. There may be a removal below $l_y(v')$ in L, but it cannot affect the half line above $l_y(v)$. Now let $p = l(v')$ and $v' \in X(v)$. Suppose, that there is an element $q \neq d$ below p in L^* and let w be the element with $u(w) = q$. In this case v' joins the chain of the w and q is removed from L^*. Obviously, the same action takes place on L. If there is only the dummy element below p on L^*, then v' opens a new chain for the restricted input. On L there may be an element below p. Nevertheless, the y-coordinate of this element has to be smaller than $l_y(v)$ and the changes on L will not affect the half line above $l_y(v)$.

If $p = u(v')$ and $v' \notin X(v)$, then $u_x(v') < l_x(v)$ and $v' \notin X(v)$ imply that $u_y(v') < l_y(v)$. Therefore p is inserted in the half line of L below $l_y(v)$. Finally, let $p = u(v')$ and $v' \in X(v)$. We then have $u_y(v') > l_y(v)$ and p is inserted in both, L^* and L. □

By induction on the number of boxes in the input we may now assume that the chain partition $\mathcal{C}_2^*$ is optimal for $X(v)$. Choose an antichain B of the order induced on $X(v)$, such that B contains an element from each chain $C^* \in \mathcal{C}_2^*$. Since every element in the antichain $U \cup \{v\}$ is incomparable to every element in $X(v)$, we conclude, that $A = B \cup U \cup \{v\}$ is an antichain. The antichain A consists of a member of every chain of the chain partition $\mathcal{C}$, i.e., $|A| = |\mathcal{C}|$. Since $|A| \leq |\mathcal{C}|$ for every antichain A and every chain partition $\mathcal{C}$, equality can only hold if A is maximum and $\mathcal{C}$minimum. This proves

Theorem 5 *A minimum chain partition of a trapezoid order on n points, given its box representation can be computed in time $O(n \log n)$ and linear space.* □

5 Maximum antichain for trapezoid orders

We first describe the geometry of antichains in a box representation. Our algorithm for maximum weighted antichains of trapezoid orders will be based on this geometric structure rather than on duality as the algorithms presented so far. First, we need some definitions.

Define the *shadow of a point* p as the set of points in the plain dominating p, i.e., $\text{shadow}(p) = \{q : q > p\}$. The *shadow of a set of points* is the union of the shadows of the elements. A downwards *staircase* is a sequence of horizontal and vertical line segments that may be obtained as the topological boundary of the shadow of a set of points. Note that any two different points on a staircase are incomparable. If S is a staircase and $l, u: V \to \mathbb{R}^2$ a set of boxes we denote the set of elements whose box intersect S by $A(S)$, i.e., $A(S) = \{v \in V : \text{box}(v) \cap S \neq \emptyset\}$.

Lemma 6 *Let* $\mathbf{P}$ *be an order given by a box representation. If* S *is a staircase then* $A(S)$ *is an antichain. Moreover, if* A *is an antichain of* $\mathbf{P}$ *then there exists a staircase* S *such that* $A \subseteq A(S)$.

Proof: Assume that $A(S)$ is not an antichain. Then there are $v, v' \in A(S)$ with $v < v'$. Consequently, for two different points $p \in \text{box}(v) \cap S$ and $p' \in \text{box}(v') \cap S$ on the staircase we have $p < p'$. But this is impossible, as noted above.

If A is an antichain of $\mathbf{P}$, let $u(A) = \{u(v) : v \in A\}$. Let staircase S be the boundary of the shadow of $u(A)$. Now suppose that there is an element $v \in A$, such that $\text{box}(v) \cap S = \emptyset$. Since $u(v)$ must lie in the shadow of $u(A)$, it follows that $l(v)$ is contained in the shadow of $u(a)$, for some $a \in A$. By definition, $u(a) < l(v)$ and hence $a < v$ in $\mathbf{P}$, a contradiction. □

Given a weighted order $\mathbf{P}$ with a box representation we define the *weight of a staircase* S as the sum of weights of all boxes intersecting S. If S is a staircase and $p \in S$, then we refer to the part of S that is above and to the left of p as *staircase ending in* p and again its weight is the sum of weights of intersecting boxes.

The following algorithm computes an antichain of maximum weight. It uses two different data structures. The sweep line L halts at every point $l(v)$ and $u(v)$, for $v \in V$. Roughly, it contains a list of weighted markers, so that the weight of marker m is the weight of a heaviest staircase ending in m. Moreover, a heaviest staircase ending in an arbitrary point y on L can be composed by joining the vertical line segment from y to the next marker m above y with a heaviest staircase ending in m. Structure L is initialized with a dummy point d of weight 0, such that d is above all points that will ever be inserted into L. The second structure Δ contains a list of all *open* boxes, i.e., boxes which have their left sides already swept but not their right ones. The total weight of all open boxes the upper sides of which lie between points y_1 and y_2 on L with $y_1 \geq y_2$ is denoted by $\Delta(y_1, y_2)$.

MAXANTICHAIN($\mathbf{P}, l, u, w$)
for each p from left to right **do**
  $m \leftarrow$ first marker above p on L
  if $p = l(v)$ for some $v \in V$ **then**
    add $w(v)$ to all markers in interval $[l_y(v), u_y(v)]$
    insert a new item in Δ at height $u_y(v)$ with weight $w(v)$
    $m^* \leftarrow$ next marker below p on L
    while $W(m) + \Delta(m, m^*) > W(m^*)$ **do**
      remove m^* from L
      $m^* \leftarrow$ next marker below p on L
  if $p = u(v)$ for some $v \in V$ **then**
    insert a new marker m_v at p_y in L
    $W(m_v) \leftarrow W(m) + \Delta(m, m_v)$
    list$(m_v) \leftarrow$ list$(m) \cup \{p\}$
    remove item at $u_y(v)$ from Δ
$T \leftarrow$ staircase of points in list(lowest(L))
for each $v \in V$ **do**
  if v intersects T **then** $A \leftarrow A \cup \{v\}$
return A

Lemma 7 *At the end of the main loop in Algorithm MAXANTICHAIN we have the following invariant. If y is an arbitrary point on L and m the next marker above y, then a maximum weighted staircase that ends in y on L has weight $W(m)+\Delta(m, y)$.*

Proof: Let W' denote the sweep line structure and Δ' denote the open box structure before a halt of the sweep line L and let W and Δ be the pair of structures after the halt. Let m be the first marker above y on L. The shadow of list$(m) \cup \{y\}$ defines a staircase S and S_y may denote that part of S that ends in point y. We show that S_y has maximal weight $W(m) + \Delta(m, y)$.

At first, suppose the sweep line L halts at some point $l(v)$ for $v \in V$. If $y > u_y(v)$ nothing has changed. If $m > u_y(v)$ and $y < u_y(v)$ then S_y intersects the new box v and has weight $W(m) + \Delta(m, y) = W'(m) + (\Delta'(m, y) + w(v))$. This weight is maximal among all staircases ending in y by the invariance assumption. If m lies between $l_y(v)$ and $u_y(v)$ then S_y has weight $W(m) + \Delta(m, y) = (W'(m) + w(v)) + \Delta'(m, y)$ which again is maximal. If $m < l_y(v)$ then the weight of S_y is $W(m) + \Delta(m, y) = W'(m) + \Delta'(m, y)$ which is the maximum weight of a staircase that avoids v. On the other hand, if a staircase S'_y ending in y intersects v then there is a $y' > l_y(v)$, such that S'_y is composed of a staircase to y' and a vertical segment from y to y'. Let m' be the first marker above y'. Since the case $y' > l_y(v)$ was already considered and by the definition of Δ, the weight of S'_y is at most

$$W(m') + \Delta(m', y') + \Delta(y', m) + \Delta(m, y) = W(m') + \Delta(m', m) + \Delta(m, y).$$

Hence, by the condition on the removal of markers in the algorithm, the weight of S'_y is at most $W(m) + \Delta(m, y)$, the weight of S_y.

Now suppose L halts at some point $u(v)$. Since v is no longer open we have to remove $u_y(v)$ from Δ'. On the other hand, we have to maintain the invariant. Thus, a new marker m_v is inserted in L with weight $W(m_v) = W'(m) + \Delta'(m, m_v)$, where m is the next marker above $u_y(v)$. Let y be so that m_v is the next marker above y, then the weight of S_y is

$$W(m_v) + \Delta(m_v, y) = W'(m) + \Delta'(m, m_v) + \Delta'(m_v, y) = W'(m) + \Delta(m, y).$$

This weight is maximal by the invariance assumption, since no new box has to be considered. □

Theorem 8 *Let $\mathbf{P} = (V, P)$ be a trapezoid graph given by a box representation and $w: V \to \mathbb{R}$ be a weighting of $\mathbf{P}$.* MAXANTICHAIN *computes a maximally weighted antichain of $\mathbf{P}$.*

Proof: After all boxes have been swept, structure Δ is empty (i.e., there is no box left open). Hence, the theorem follows from the invariant of Lemma 7. □

L may be implemented by a balanced binary tree. One has to be careful only about adding some weight w to a whole interval $[l, u]$. Let each node of the tree have some extra field holding the increment in the weight for all nodes in its subtree. During a rebalancing rotation this fields must be corrected accordingly. But it is easily seen that only a constant number of such fields is affected. Consequently, the addition of some weight to an interval as well as insertion, deletion, predecessor and successor queries, and the computation of the weight of some element can all still be done in time $O(\log n)$.

Δ may be implemented by any one dimensional range tree where insertion, deletion, and query again takes $O(\log n)$ time. The main loop is executed n times and each step therein takes logarithmic time save the while loop. But in total the while loop is executed at most n times since each removed point must have been inserted before. Of course, the test for intersection of a box v with staircase T can be done in time $O(\log n)$. In summary, we obtain

Theorem 9 *A maximum weighted antichain of a trapezoid order on n points, given its box representation, can be computed in time $O(n \log n)$ and linear space.* □

Note that one can do without the Δ structure if one uses subtraction in the W structure. But the above algorithm is easier to understand and it can be adapted to the case where no subtraction is allowed (e.g., in semigroups).

We conclude this section with an open problem. We have given optimal algorithms for the classical chain and antichain problems for trapezoid orders. Also, for k-trapezoidal orders $k \geq 3$ we have obtained a fast algorithm for maximum weighted chain. Is there an algorithm for maximum antichain for k-trapezoidal orders $k \geq 3$ whose running time improves over the complexity of bipartite matching and hence the complexity of the algorithm for general orders?

6 Algorithms for circle trapezoid graphs

A *circle trapezoid* is the region in a circle that lies between two non-crossing chords and CT-*graphs* are the intersection graphs of families of circle trapezoids on a common circle. Figure 2 gives an example. In this section we develop polynomial algorithms for the maximum weighted clique and a maximum weighted independent set problems on CT-graphs.

6.1 Crossing graphs and independent sets of CT-graphs

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a CT-graph, of course, we will assume that a representation of $\mathbf{G}$ is given. Let p be an arbitrary point on the circle and let C_p be the set of vertices of $\mathbf{G}$ whose circle trapezoid contains p. Note that C_p induces a clique of $\mathbf{G}$, therefore, an independent set of $\mathbf{G}$ can contain at most one element from C_p. Using p as the 'origin' of the circle and fixing an orientation (clockwise) of the circle we can define a unique representation for circle trapezoids. The representation consists of 5-tuple $(t_1, t_2, t_3, t_4, \sigma)$. The first four components are the corners of the circle trapezoid in clockwise order starting from p. The fifth component σ is a sign, $+$ or $-$, where $+$ indicates that p is contained in one of the arcs of the circle trapezoid.

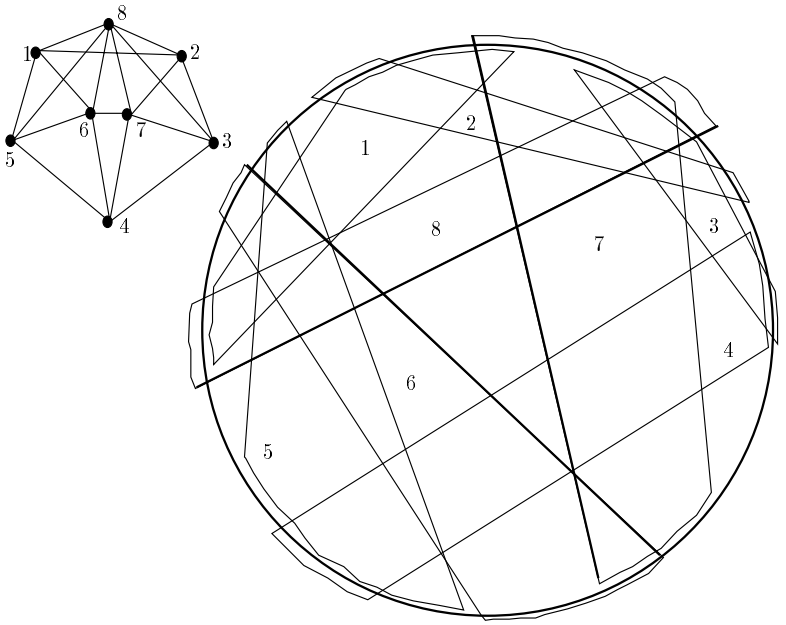

Figure 2. A circle trapezoid graph $\mathbf{G}$ with a representation

Define a *double interval* as a pair (I_1, I_2) of intervals on the real line, where I_2 is a subinterval of I_1, i.e., $I_2 \subset I_1$. Let $I = (I_1, I_2)$ and $J = (J_1, J_2)$ be double intervals. We say I *contains* J if $J_1 \subset I_2$ and call them *disjoint* if $I_1 \cap J_1 = \emptyset$. Two double intervals are called *crossing* if they are not disjoint and non of them is contained in the other. Call a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ a *crossing graph* if its vertices can be put in one to one correspondence to a collection of double intervals such that two vertices of $\mathbf{G}$

are adjacent if and only if their corresponding double intervals cross. It is not hard to see that the class of crossing graphs contains both, trapezoid graphs and overlap graphs (recall that a graph is an overlap graph if and only if it is a circle graph).

Our next lemma relates CT-graphs and crossing graphs.

Lemma 10 *Let* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *be a CT-graph given by a representation and* C_p *be the set of all vertices of* $\mathbf{G}$ *whose circular trapezoid share a specified point* p *on the circle.*

For a subset W *of* $V \setminus C_p$ *the subgraph of* $\mathbf{G}$ *induced by* W *is a crossing graph.*

Proof: Given the circular trapezoid representation we associate to a vertex $v \in W$ with circular trapezoid $(t_1, t_2, t_3, t_4, -)$ two arcs along the circle. A_1 is the arc from t_1 to t_4 and A_2 is the arc from t_2 to t_3, in both cases we choose the arc which does not contain p. Obviously, $A_2 \subset A_1$. Cutting the circle at p we obtain a line with a collection of double intervals representing the subgraph of $\mathbf{G}$ induced by W. □

For $v \in V$ let $N[v]$ denote the set of neighbors of v together with v itself and let $\mathbf{G}(v)$ be the subgraph of $\mathbf{G}$ induced by $V \setminus N[v]$. Also, let $\mathbf{G}_p$ denote the subgraph induced by $V \setminus C_p$. We have remarked earlier that the vertices of C_p form a clique in $\mathbf{G}$. Therefore a maximum independent set I of $\mathbf{G}$ is either a maximum independent set in $\mathbf{G}_p$ or there is a $v \in C_p$, such that $I = I' \cup \{v\}$ where I' is a maximum independent set of $\mathbf{G}(v)$. Since $C_p \subseteq N[v]$ for all $v \in C_p$ the lemma shows that each of the above graphs $\mathbf{G}(v)$, as well as $\mathbf{G}_p$ are crossing graphs. This reduces the detection of a maximum independent set of a CT-graph to at most n maximum independent set problems on crossing graphs. We therefore turn the attention to the maximum independent set problem for crossing graphs. Our algorithm for this problem is very much alike the algorithm given by Gavril [Gav] (see also Golumbic [Gol]) for the case of overlap graphs.

For a pair of double intervals we have defined the relations containment, disjointness and crossing and by definition every pair of double intervals is in exactly one of these relations. The containment is a antisymmetric and transitive relation, i.e., an order relation. For the disjointness we only need the first interval of each double interval, therefore, we can transitively orient disjoint pairs by the relation 'lies entirely to the left', this gives an interval order.

To compute the maximum independent set of a crossing graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ given by a family $\mathcal{I}$ of double intervals we proceed as follows. First, the containment order $\mathbf{P} = (V, P)$ and the interval order $\mathbf{Q} = (V, Q)$ corresponding to $\mathcal{I}$ are extracted and a linear extension $L = v_1, \ldots, v_n$ of $\mathbf{P}$ is computed. We artificially extend $\mathbf{P}$ and L by an element v_{n+1} of weight 0, such that $v_{n+1} > v_i$ for all $i = 1, \ldots, n$. This preprocessing can be accomplished in time $O(n^2)$. Next, the following algorithm is called.

MAXINDEPENDENTSET($\mathbf{P}, \mathbf{Q}, L$)
for $i = 1$ to $n+1$ **do**
  $U_i \leftarrow \{v_j : v_j > v_i \text{ in } P\}$
  $C \leftarrow$ maximum W-weighted $\mathbf{Q}$-chain of elements of U_i
  $W(v_i) \leftarrow w(v_i) + \sum_{v \in C} W(v)$
  $I(v_i) \leftarrow \{v_i\} \cup \bigcup_{v \in C} I(v)$
return $I(v_{n+1})$

It is important to note, that U_i only contains elements v_j with $j > i$, hence, the weights $W(v_j)$ of all elements in U_i have already been computed before the ith round. The following invariance of the algorithm is easily proved. *At the end of the ith round for all $j \leq i$ the weight $W(v_j)$ is the weight of a maximum independent set $I(v_j)$ containing only elements $v \in U_j \cup \{v_j\}$, i.e., elements with $v \leq v_j$ in* $\mathbf{P}$. From this invariant $I(v_{n+1})$ is a maximum independent set for $\mathbf{G}$.

Clearly, every but the second instruction in the loop can be executed in $O(n)$ time. The second instruction itselve is a maximum chain computation in an interval order. This problem can be solved in linear time when the endpoints of the intervals are available in increasing order. For completeness we sketch an algorithm for this problem. Visit the endpoints from left to right and maintain the weight ρ of the longest chain among intervals whose right endpoint has already been seen. When reaching the left endpoint of an interval, say the interval of v, we know that the maximum weighted chain having v as maximal element has weight $W(v) = \rho + w(v)$. At the right endpoint of v's interval we update ρ by the rule $\rho = \max\{\rho, W(v)\}$. Note that this algorithm can be seen as an one dimensional version of algorithm MAXCHAIN in Section 2, i.e., instead of a sweep line we use a sweep point and can thus save the search for the relevant marker.

Lemma 11 *Algorithm MAXINDEPENDENTSET solves the maximum weighted independent set problem for crossing graphs in $O(n^2)$.* □

Recall that the solution for the maximum independent set problem for CT-graphs is a either a maximum independent set in $\mathbf{G}_p$ or one of the sets $I = I' \cup \{v\}$ where $v \in C_p$ and I' is a maximum independent set in $\mathbf{G}(v)$. We show next that having applied algorithm MAXINDEPENDENTSET to $\mathbf{G}_p$ the problem for a each of the graphs $\mathbf{G}(v)$, $v \in C_p$, can be solved by a single maximum chain computation in an interval order, i.e., in $O(n)$ time.

Let $v \in C_p$ and let the circular trapezoid of v be given by $(s_1, s_2, s_3, s_4, +)$. The double intervals corresponding to vertices of $\mathbf{G}(v)$ are exactly those with $I_1 \subset (s_1, s_2)$ or $I_1 \subset (s_3, s_4)$. Let v_i be an element of $\mathbf{G}(v)$ and recall that the set U_i is the set of elements whose double interval is contained in the double interval of v_i. It follows that U_i is contained in $\mathbf{G}(v)$ and hence that sets $I(v_i)$ and weights $w(v_i)$ computed by MAXINDEPENDENTSET with input $\mathbf{G}_p$ and with input $\mathbf{G}(v)$ are equal. To solve the problem for $\mathbf{G}(v)$ it thus suffices to select the intervals contained in (s_1, s_2) or (s_3, s_4) and compute a maximum weighted chain of this set of intervals.

Theorem 12 *The maximum weighted independent set problem for CT-graphs can be solved in* $O(n^2)$. □

6.2 Cliques of CT-graphs

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a CT-graph, given by a circular trapezoid representation. A clique C of $\mathbf{G}$ is called a *Helly-clique* with respect to the representation, if there is a point p in the interior of the circle, such that the circle trapezoid of every vertex $v \in C$ contains p. Our algorithm for the maximum clique problem on CT-graphs proceeds in two stages. In the first stage it determines a maximum Helly-clique of $\mathbf{G}$, in the second stage a maximum non-Helly-clique, NH-*clique* for short, is computed. The larger of the two cliques is a maximum clique for $\mathbf{G}$.

The determination of a maximum Helly-clique is a purely geometrical problem. Consider the set of $2n$ chords of the circular trapezoids. These chords cut the interior of the circle into regions. We define the weight of a region as the number of circular trapezoids that contain it. Impose an orientation on each chord, such that the interior of the trapezoid it bounds is to the left of the chord when looking in positive direction. Assuming that the chords are in general position we find four regions in the neighborhood of every crossing of chords. Given the weight of one of these regions and the orientation of the chords, we can easily determine the weights of the other three regions. Simultaneously we can compute the actual set of vertices whose circular trapezoids contain a region. It follows, that a sweep line that halts at every intersection of chords can maintain the weights (and the corresponding sets of vertices) of the regions it intersects. The update at every crossing can be done in constant time and there are less than $4n^2$ crossings. Hence, the complexity of the algorithm is dominated by the sorting of the crossings. We summarize: the maximum Helly-clique can be found in $O(n^2 \log n)$.

We now turn to the case of NH-cliques. A NH-*triangle* T in $\mathbf{G}$ is a three element clique, such that the circular trapezoids of the three vertices have no point in the circle in common. Let $T = \{x, y, z\}$ be a NH-triangle of $\mathbf{G}$ and for $v \in V$ let $B(v)$ be the circular trapezoid of v. Let c be one of the chords bounding $B(x)$, we call c the *inner chord* of x if $B(x)$ is on one side of c and $B(y) \cap B(z)$ is on the other side of c. The side of c not containing $B(x)$ is the *outer side* of x. The intersection of the outer sides of the elements of a NH-triangle T is the *enclosed area* of T, of course, the enclosed area is a triangle in the interior of the circle. The definitions are illustrated in Figure 2, the set $\{6, 7, 8\}$ is a NH-triangle, the outer sides of their circular trapezoids are bold.

A family $\mathcal{F}$ of subsets of a set X has *Helly number* k if for every subfamily $\mathcal{S}$ of $\mathcal{F}$ the property that any k members of $\mathcal{S}$ have non-empty intersection implies that the intersection over all members of $\mathcal{S}$ is non-empty. It is well known, that the Helly number of convex objects in the plane is 3, see [Gol]. As a consequence we obtain.

Lemma 13 *Every NH-clique in a CT-graph contains a NH-triangle.* □

Based on this lemma an algorithm to compute a maximum NH-clique in a CT-graph $\mathbf{G}$ can proceed as follows. Enumerate all NH-triangles in $\mathbf{G}$ and for each NH-triangle T compute a maximum clique among the cliques containing T. Our approach is only slightly more sophisticated. A NH-triangle T is called a *maximal triangle* for a clique C, if the enclosed area of T is maximal among the enclosed areas of NH-subtriangles of C, i.e., there is no NH-subtriangle T' of C, whose enclosed area contains the enclosed area of T.

In what follows we show how a maximum clique among the cliques containing a given NH-triangle T as maximal triangle can be computed.

Lemma 14 *If C is a clique in $\mathbf{G}$ containing a NH-triangle T as a maximal triangle, then every $v \in C$ has the property that the intersection of $B(v)$ with each of the inner chords c_x for $x \in T$ is non-empty.*

Proof: Suppose $B(v) \cap c_x = \emptyset$, then $B(v)$ is completely on one side of c_x. Since $B(v) \cap B(x) \neq \emptyset$ we conclude that $B(v)$ cannot be on the outer side of c_x. It follows that $T' = T \cup \{v\} \setminus \{x\}$ is a NH-triangle contained in C and the enclosed area of T' contains the enclosed area of T. This contradicts the assumption that T is a maximal triangle for C. □

For a NH-triangle T let $V(T)$ be the set of vertices $v \in V$ such that the intersection of $B(v)$ with each of the inner chords c_x for $x \in T$ is non-empty. As the previous lemma shows, a clique of $\mathbf{G}$ containing T as a maximal triangle is a subset of $V(T)$.

The six endpoints of the inner chords of the elements of T partition the circle into six disjoint arcs $A_1, \ldots, A_6$. The numbering of these arcs is assumed to be consecutive. We call the arcs A_i and A_{i+3} for $i = 1, 2, 3$ a pair of *opposite* arcs. An easy case analysis proves the next lemma.

Lemma 15 *If $v \in V(T)$, then there is at least one $i \in \{1, 2, 3\}$, such that the intersection of $B(v)$ with both of A_i and A_{i+3} is non-empty.* □

Let $f : V(T) \to \{1, 2, 3\}$ be a function such that $B(v) \cap A_{f(v)} \neq \emptyset$ and $B(v) \cap A_{f(v)+3} \neq \emptyset$ for every $v \in V(T)$. Also let $V_i(T) = \{v \in V(T) : f(v) = i\}$ for $i = 1, 2, 3$.

Lemma 16 *If $i, j \in \{1, 2, 3\}$ with $i \neq j$ then for all $v_i \in V_i(T)$ and $v_j \in V_j(T)$ the circular trapezoids $B(v_i)$ and $B(v_j)$ intersect, i.e., v_i and v_j are adjacent in $\mathbf{G}$.* □

As a consequence we obtain that, if C_i is a clique in the subgraph $\mathbf{G}_i(T)$ of $\mathbf{G}$ induced by $V_i(T)$ for $i = 1, 2, 3$, then $T \cup C_1 \cup C_2 \cup C_3$ is a clique in $\mathbf{G}$. Moreover, and more important.

Lemma 17 *C is a maximum clique containing T as a maximal triangle, exactly if $C_i = C \cap V_i(T)$ is a maximum clique in $\mathbf{G}_i(T)$ for each $i = 1, 2, 3$.* □

We next show that the problem of finding a maximum clique in $\mathbf{G}_i(T)$ for $i = 1,2,3$ is a maximum clique problem in a trapezoid graph. By symmetry we may restrict our attention to $\mathbf{G}_1(T)$. When speaking of an arc a, b on the circle we henceforth mean that a and b are points on the circle and the arc emanates from a turning clockwise to b.

Let A_1 be the arc a_1, a_2 and A_4 be the arc a_3, a_4. We first consider two easy cases.

Case 1. If $v \in V_1(T)$ is such that $B(v)$ contains either A_1 or A_4, then v is adjacent to every element in $V_1(T) \setminus \{v\}$ and hence contained in every maximal clique of $\mathbf{G}_1(T)$. $\triangle$

Case 2. If $v \in V_1(T)$ is such that $B(v)$ contains a_1 and a_3 or a_2 and a_4, then v is adjacent to every element in $V_1(T) \setminus \{v\}$ and hence contained in every maximal clique of $\mathbf{G}_1(T)$. $\triangle$

If $v \in V_1(T)$ is such that either Case 1 or Case 2 applies, then define $B^*(v)$ as the circular trapezoid spanned by the arcs A_1 and A_4.

We now suppose that $v \in V_1(T)$ is an element, such that neither Case 1 nor Case 2 applies to v.

Lemma 18 *One of the two chords at the border of $B(v)$ joins a point in A_1 with a point in A_4.* $\square$

Let $B(v)$ be the convex hull of the arcs t_1, t_2 and t_3, t_4. We may assume that at least one of t_1 and t_2 is in A_1.

First suppose, that $t_2 \in A_1$ and $t_3 \in A_4$, i.e., the chord t_2, t_3 joins a point in A_1 with a point in A_4. Let t_4' be the first of a_4 and t_4, when looking from t_3 in clockwise direction, also let t_1' be the first of a_1 and t_1, when looking from t_2 in counterclockwise direction. Define $B^*(v)$ as the circular trapezoid spanned by the arcs t_1', t_2 and t_3, t_4'.

If $t_1 \in A_1$ and $t_4 \in A_4$, then we define t_2' as the first of a_2 and t_2 in clockwise direction from t_1 and t_3' as the first of a_3 and t_3 in counterclockwise direction from t_4. Define $B^*(v)$ as the circular trapezoid spanned by the arcs t_1, t_2' and t_3', t_4.

Lemma 19 $\mathbf{G}_1(T)$ *is the intersection graph of the family of circular trapezoids $B^*(v)$ for $v \in V_1(T)$.* $\square$

The circular trapezoid $B^*(v)$ is spanned by one arc contained in A_1 and one arc in A_4 for every $v \in V_1(T)$. This obviously implies that the graph $\mathbf{G}_1(T)$ a trapezoid graph. We remark that a box representation of $\mathbf{G}_1(T)$ can be produced in $O(n)$. Hence the maximum clique problem for $\mathbf{G}_1(T)$ can be solved by the algorithm of Section 4 (as remarked earlier the of equal coordinates in the set $\{l(v), u(v) : v \in V\}$ can be made different easily).

Let us review the main steps of the a polynomial algorithm for the maximum clique problem of CT-graphs. First, the algorithm searches for a maximum Helly-clique in $\mathbf{G}$. Then, for every NH-triangle T in $\mathbf{G}$ a maximum clique C with the property that T is a maximal triangle in C is found by solving three maximum clique

instances for trapezoid graphs. A largest among the cliques found in the second step and the clique of step one is the solution for the maximum clique problem on **G**. In the analysis of the time complexity of the algorithm we have to count $O(n \log n)$ for the computation of a maximum clique containing a NH-triangle T as a maximal triangle. Unfortunately there can be as much as $O(n^3)$ NH-triangles in a representation of **G**. We summarize the results of this subsection.

Theorem 20 *The maximum weighted clique problem for CT-graphs can be solved in* $O(n^4 \log n)$. □

7 Conclusion

We have shown how the box representation, the sweep line paradigm and the analogy with 2-dimensional orders may help design algorithms for trapezoid orders. In particular we have obtained optimal $O(n \log n)$ algorithms for the four problems considered. It remains to investigate other problems, e.g, domination problems, on the class of trapezoid graphs.

The new class of circular trapezoid graphs has been introduced. The maximum clique and maximum independent set problems on this large class have been shown to be polinomially solvable. Trapezoid graph algorithms are important subroutines for these algorithms. Similarly, they may serve as subroutines for algorithms to compute parameters of tolerance graphs.

References

[ABMP] H. Alt, N. Blum, K. Mehlhorn and M. Paul, Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}(m/\log n)^{0.5})$ *Inf. Proc. Letters* **37** (1991), 237–240.

[CLR] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, (1989).

[DGP] I. Dagan, M.C. Golumbic and R.Y. Pinter, Trapezoid Graphs and their Coloring, *Discr. Appl. Math.* **21** (1988) 35–46.

[Fre] M.L. Fredman, On Computing the Length of Longest Increasing Subsequences, *Discr. Math.* **11** (1975), 29–35.

[Gav] F. Gavril, Algorithms for a Maximum Clique and a Maximum Independent Set of a Circle Graph, *Networks* **3** (1973), 361–273.

[Gol] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[Gol] E. Helly, Über die Menge konvexer Körper mit gemeinsamen Punkten, *Jahresb. d. Dt. Mathem. Ver.* **32** (1923).

[MCSp] R. McConnel and J. Spinrad, Linear-Time Modular Decomposition and Efficient Transitive Orientation of Undirected Graphs, *Proc. 5. Annual Symp. on Discr. Alg.*, (1994).

[PaSt] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, (1983).

[Smi] M. Smid, Range trees with slack parameter, *Algorithms Review*, **2(2)**, (1991), 77–87.

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