Independent Set of Intersection Graphs of Convex Objects in 2D *

Pankaj K. Agarwal Nabil H. Mustafa

Department of Computer Science,
Duke University, Durham, NC 27708-0129, USA.
{pankaj, nabil}@cs.duke.edu

Abstract

The *intersection graph* of a set of geometric objects is defined as a graph $G = (S, E)$ in which there is an edge between two nodes $s_i, s_j \in S$ if $s_i \cap s_j \neq \emptyset$. The problem of computing a maximum independent set in the intersection graph of a set of objects is known to be NP-complete for most cases in two and higher dimensions. We present approximation algorithms for computing a maximum independent set of intersection graphs of convex objects in $\mathbb{R}^2$. Specifically, given (i) a set of n line segments in the plane with maximum independent set of size α, we present algorithms that find an independent set of size at least $(\alpha/2\log(2n/\alpha))^{1/2}$ in time $O(n^3)$ and $(\alpha/2\log(2n/\alpha))^{1/4}$ in time $O(n^{4/3}\log^c n)$, (ii) a set of n convex objects with maximum independent set of size α, we present an algorithm that finds an independent set of size at least $(\alpha/2\log(2n/\alpha))^{1/3}$ in time $O(n^3 + \tau(S))$, assuming that S can be preprocessed in time $\tau(S)$ to answer certain primitive operations on these convex sets, and (iii) a set of n rectangles with maximum independent set of size βn, for $\beta \leq 1$, we present an algorithm that computes an independent set of size $\Omega(\beta^2 n)$. All our algorithms use the notion of *partial orders* that exploit the geometric structure of the convex objects.

1 Introduction

An *independent set* of a graph is a subset of pairwise nonadjacent nodes of the graph. The maximum-independent-set problem asks for computing a largest independent set of a given graph. Given a graph G and an integer $k > 0$, determining whether there is an independent set in G of size k is known to be NP-complete even for many restricted cases (e.g. planar graphs [12], bounded-degree graphs [21], geometric graphs [22]). Naturally, the attention then turned toward *approximating* the largest independent set in polynomial time. Unfortunately, the existence of polynomial-time algorithms for approximating the maximum independent set efficiently for general graphs is unlikely [14]. However, efficient approximation algorithms are known for many restricted classes of graphs. For planar graphs, approximation algorithms exist that can compute an independent set of

*Research is supported by NSF grants ITR-333-1050, EIA-98-70724, EIA-01-31905, and CCR-00-86013.

size arbitrarily close to the size of the maximum independent set. Note that a graph is planar if and only if there exists a set of unit disks in the plane whose contacts give the edges of the planar graph [19]. Thus a natural direction is to investigate the independent-set problem for the graphs induced by a set of geometric objects. The intriguing question there is whether (and what) geometric nature of objects aids in efficient computation of maximum independent set. One such family of graphs arising from geometric objects that have been studied are the so-called intersection graphs.

Given a set $S = \{s_1, \ldots, s_n\}$ of geometric objects in $\mathbb{R}^d$, the *intersection graph* of S, $G_S = (V, E)$ is defined as follows: each node $v_i \in V$ corresponds to the object s_i and $e_{ij} \in E$ if $s_i \cap s_j \neq \emptyset$. A subset $V' \subseteq V$ is an independent set in G_S if for every pair of nodes $v_i, v_j \in S'$, $s_i \cap s_j = \emptyset$. For brevity, we say "independent set of S" when we mean "independent set of the intersection graph of S". In this paper, we present approximation algorithms for the independent-set problem on intersection graphs of line segments and convex objects in the plane.

Besides the inherent interest mentioned above, independent sets of intersection graphs have found applications in map labeling in computational cartography [2], and frequency assignment in cellular networks [18]. For example, in the map-labeling problem, we are given a set of labels of geometric objects, and the goal is to place the maximum number of labels that are pairwise disjoint. Computing the maximum independent set of these labels yields a labeling with the maximum number of labeled objects.

Related work. Given S, let $\mathcal{I}^*(S)$ denote a maximum independent set of G_S, and let $\alpha(S)$ denote it size. We will use α to denote $\alpha(S)$ when S is clear from the context. We say that an algorithm computes a c-approximation to $\mathcal{I}^*(S)$ if it computes an independent set of size at least $\alpha(S)/c$.

For a general graph $G(V, E)$ with n vertices, there cannot be a polynomial-time approximation algorithm with approximation ratio better than $n^{1-\epsilon}$ for any $\epsilon > 0$ unless NP = ZPP [14]. Currently the best algorithm for a general graph finds an independent set of size $\Omega(\alpha \cdot \log^2 n/n)$ [7], where α is the size of a maximum independent set in G.

However, for intersection graphs of geometric objects, better approximation ratios are possible. If I is a set of intervals in $\mathbb{R}$, then the maximum independent set of the intersection graph of I can be computed in linear time. Computing $\mathcal{I}^*(S)$ is known to be NP-complete if S is a set of unit disks or a set of orthogonal segments in $\mathbb{R}^2$ [16]. For unit disks in $\mathbb{R}^2$, a polynomial time $(1+\epsilon)$-approximation scheme was proposed in [15]. For arbitrary disks, independently Erlebach *et al.* [11] and Chan [8] presented a polynomial time $(1+\epsilon)$-approximation scheme. The above schemes for computing independent set of disks use *shifted dissection*, which relies heavily on the fact that the geometric objects are disks (or "fat" objects). A divide-and-conquer technique is used for the case of the intersection graphs of axis-parallel rectangles in the plane, for which Agarwal *et al.* [2] presented a $O(\log n)$-approximation algorithm in time $O(n \log n)$. If the rectangles have unit height, they describe a $(1+\epsilon)$-approximation scheme with running time $O(n \log n + n^{2/\epsilon - 1})$. Berman *et al.* [5] show that a $\log_k n$-approximation can be computed in $O(n^k \alpha(S))$ time. Recently Chan [9] improved these algorithms by describing an algorithm that returns a $\log_k \alpha(S)$-approximation in time $O(n \log n + n\omega^{k-1})$, where $k \geq 2$ is any constant. Efficient algorithms are known for other classes of graphs as well [3, 6].

In other related work [17], it was shown that the problem of recognizing intersection graphs of

line segments, i.e., given a graph G, does there exist a set of segments whose intersection graph is G, is NP-hard. Another work related to ours is of Pach and Tardos [20]. Given a set of disjoint objects (which could be line segments or convex shapes) in $\mathbb{R}^2$ lying on a sheet of glass, they study the following combinatorial question: by iteratively cutting the glass into two separate sheets with a straight line, and then recursively the cutting the two pieces, how many objects can be separated into different sheets of glass? Using decomposition schemes similar to ours, they present various upper and lower bounds on the number of objects that can be separated.

Our results. We first present approximation algorithms for a set S of n segments in $\mathbb{R}^2$, all of which intersect a common vertical line. We show that we can compute [1]

- in $O(n^3)$ time an independent set of S of size at least $\sqrt{\alpha}$, and
- in $O(n^{4/3}\log^c n)$ time an independent set of S of size at least $\alpha^{1/4}$.

Using these results, we show that for an arbitrary set S of segments in $\mathbb{R}^2$, we can compute an independent set of size at least

- $\sqrt{\alpha/(2\log(2n/\alpha))}$ in time $O(n^3)$, or
- $(\alpha/(2\log(2n/\alpha)))^{1/4}$ in time $O(n^{4/3}\log^c n)$.

We then extend our results to convex sets. Namely, for a family S of n convex sets in $\mathbb{R}^2$, we can compute in $O(n^3+\tau(S))$ time an independent set of size at least $(\alpha/2\log(2n/\alpha))^{1/3}$, assuming that certain primitive operations (namely sidedness and pairwise object intersection queries) on these convex sets can be performed by preprocessing S in $\tau(S)$ time. Finally, for a set of n rectangles with maximum independent set of size βn, for some $\beta \leq 1$, we compute in $O(n^3)$ time, an independent set of size at least $\Omega(\beta^2 n)$.

Organization. In Section 2 we describe approximation algorithms for a set of line segments in $\mathbb{R}^2$. Section 3 describes the algorithm for convex sets, and Section 4 presents approximation algorithms for the case of axis-parallel rectangles.

2 Approximation Algorithms for Line Segments

Let $S=\{s_1,\ldots,s_n\}$ be a set of line segments in $\mathbb{R}^2$. Let $x(p), y(p)$ denote the x- and y-coordinates of a point $p\in\mathbb{R}^2$. Let $l(s)$ (resp. $r(s)$) denote the x-coordinate of the left (resp. right) endpoint of the segment $s\in S$, and let σ_i denote the slope of s_i.

2.1 A $\sqrt{\alpha}$-Approximation Algorithm

In this section we assume that all the segments in S intersect the y-axis. We also assume that the segments in S are sorted in increasing order of their intersection points with the y-axis, and we use

[1] All logarithms in this paper are base 2.

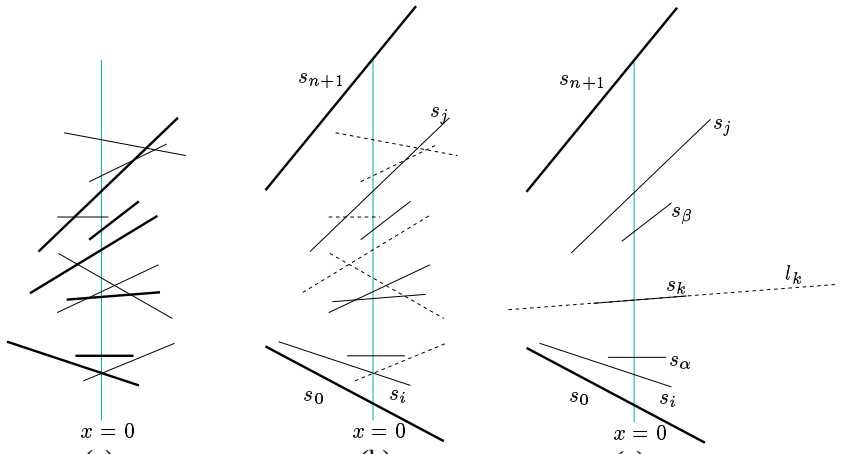


Figure 1: (a) Bold segments form an increasing s-monotone sequence, (b) S_{ij} in solid (c) s_i, s_j, s_k as in the proof of Lemma 2.2.

$S = \langle s_1, \ldots, s_n \rangle$ to denote this sorted sequence.

We call a subsequence $S' = \langle s_{i_1}, \ldots, s_{i_m} \rangle$ of S *s-monotone* (see Figure 1(a)) if

- $s_{i_j} \cap s_{i_k} = \emptyset$ for all $1 \leq j < k \leq m$.
- $\sigma_{i_j} < \sigma_{i_{j+1}}$ for all $1 \leq j < m$ (called *increasing* s-monotone) *or* $\sigma_{i_j} > \sigma_{i_{j+1}}$ for all $1 \leq j < m$ (called *decreasing* s-monotone).

Lemma 2.1 *Let $I \subseteq S$ be a subsequence of pairwise-disjoint segments. There exists an s-monotone sequence $I' \subseteq I$ of size at least $\sqrt{|I|}$.*

PROOF. By Dilworth's theorem [10], there is a subsequence I' of I such that the slopes of the segments are either monotonically increasing or monotonically decreasing, and the size of I' is at least $\sqrt{|I|}$. Since the segments in I are pairwise disjoint, I' is s-monotone. □

We describe an algorithm for computing the longest s-monotone subsequence of S. By Lemma 2.1, its size is at least $\sqrt{\alpha(S)}$. Without loss of generality, we describe how to compute the longest increasing s-monotone subsequence; the same procedure can compute the longest decreasing s-monotone subsequence of S, and we return the longer of the two.

We add a segment s_0 to S such that it intersects y-axis below all the segments of S, does not intersect any segment of S, $\sigma_0 < \sigma_i$ for all $i \geq 1$, and it spans all the other segments of S (i.e., $l(s_0) < l(s_i)$ and $r(s_0) > r(s_i)$ for all $1 \leq i \leq n$). We add another similar segment s_{n+1} that intersects the y-axis above all the other segments in S, does not intersect any segment in S, and $\sigma_{n+1} > \sigma_i$ for all $i \leq n$.

For $0 \leq i < j \leq n+1$ such that $s_i \cap s_j = \emptyset$ and $\sigma_i < \sigma_j$, let $S_{ij} \subseteq S$ denote the subsequence of segments s_k s.t.

(S1) $i < k < j$,

(S2) $\sigma_i < \sigma_k < \sigma_j$,

(S3) $l(s_k) > \max\{l(s_i), l(s_j)\}$,

(S4) $s_i \cap s_k = \emptyset$ and $s_j \cap s_k = \emptyset$.

See Figure 1(b) for an illustration of S_{ij}. Let $\Phi(i,j) \subseteq S_{ij}$ denote the longest increasing s-monotone subsequence of S_{ij}. If there is more than one such sequence, we choose the lexicographically minimum one. Set $\phi(i,j) = |\Phi(i,j)|$. We wish to compute $\Phi(0, n+1)$. Note that by definition of s_0 and s_{n+1}, $S_{0(n+1)} = S$.

Lemma 2.2 *For all* $0 \leq i < j \leq n+1$,

$$\phi(i,j) = \max_{s_k \in S_{ij}} \phi(i,k) + \phi(k,j) + 1. \tag{1}$$

PROOF. Let $\Phi(i,j) = \langle s_{a_1}, \ldots, s_{a_u} \rangle$. Let s_{a_h} be the segment in $\Phi(i,j)$ with the leftmost left endpoint, i.e., $l(s_{a_h}) \leq l(s_{a_k})$ for all $1 \leq k \leq u$. Note that $\langle s_{a_1}, \ldots, s_{a_{h-1}} \rangle \subseteq S_{ia_h}$ and $\langle s_{a_{h+1}}, \ldots, s_{a_u} \rangle \subseteq S_{a_h j}$. Since each of these two subsequences is s-monotone and $s_{a_h} \in S_{ij}$, $\phi(i,a_h) \geq h-1$ and $\phi(a_h,j) \geq u-h$. Therefore $\phi(i,j) = \phi(i,a_h) + \phi(a_h,j) + 1$ and hence

$$\phi(i,j) \leq \max_{s_k \in S_{ij}} \phi(i,k) + \phi(k,j) + 1.$$

Conversely, let $s_k \in S_{ij}$. By definition of S_{ik} and S_{kj} (cf. $(S1)$–$(S4)$), for all $s_\alpha \in S_{ik}$ and $s_\beta \in S_{kj}$,

(i) $i < \alpha < k < \beta < j$,

(ii) $\sigma_i < \sigma_\alpha < \sigma_k < \sigma_\beta < \sigma_j$,

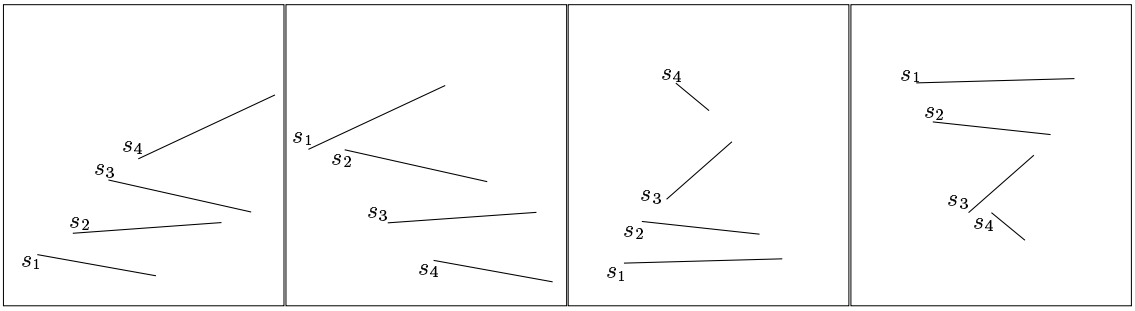
(iii) $l(s_\alpha),\ l(s_\beta) > l(s_k) > \max\{l(s_i),\ l(s_j)\}$,

(iv) $s_\alpha \cap s_k = \emptyset$, and $s_\beta \cap s_k = \emptyset$.

See Figure 1(c). As observed in [20], (i)–(iv) imply that the line l_k supporting s_k does not intersect s_α and s_β. Indeed, (i) & (ii) imply that l_k does not intersect s_α or s_β to the right of the y-axis, and (iii) & (iv) imply that l_k does not intersect s_α or s_β to the left of the y-axis. Since s_α and s_β lie on the opposite sides of l_k, they neither intersect each other, nor s_i or s_j. Hence, the segments in $\Phi(i,k) \cup \Phi(k,j)$ are pairwise disjoint. Moreover, the fact that s_α and s_β do not intersect s_i or s_j and (i)–(iv) imply that s_α and s_β satisfy $(S1)$–$(S4)$ for S_{ij}. Hence $S_{ik} \cup S_{kj} \subseteq S_{ij}$.

Therefore the sequence $\langle \Phi(i,k) \circ \langle s_k \rangle \circ \Phi(k,j) \rangle$ is an s-monotone subsequence of S_{ij}. Hence,

$$\phi(i,j) \geq \phi(i,k) + \phi(k,j) + 1.$$



(a) (b) (c) (d)

Figure 2: The four types of l-monotone sequences described in Lemma 2.4.

This completes the proof of the lemma. □

We can compute $\phi(i,j)$ using a dynamic-programming approach. We can compute the set S_{ij} in $O(n)$ time and $\phi(i,j)$ in another $O(n)$ time, assuming $\phi(i,k)$ and $\phi(k,j)$ have already been computed. Therefore, the total time spent in computing $\phi(i,j)$ for all $1 \leq i < j \leq (n+1)$ is $O(n^3)$. Putting everything together, we conclude the following.

Theorem 2.3 *Given a set S of n segments in $\mathbb{R}^2$, all intersecting a common vertical line, an independent set of size at least $\sqrt{\alpha(S)}$ can be computed in time $O(n^3)$.*

2.2 A $\alpha^{3/4}$-Approximation Algorithm

We now present a faster algorithm at the expense of a larger approximation factor. The algorithm again tries to find a large subset of $\mathcal{I}^*(S)$ that has a certain special structure, which allows its computation in polynomial time. We assume that all the segments of S intersect the y-axis and are sorted in increasing order by the x-coordinates of their left endpoints. Let $S = \langle s_1, \ldots, s_n \rangle$ denote this sequence. Let c_i be the y-coordinate of the intersection point of s_i with the y-axis.

Lemma 2.4 *Let $I \subseteq S$ be a subsequence of pairwise-disjoint segments. Then there exists a subsequence $I'' = \langle s_1, \ldots, s_m \rangle$, such that $I'' \subseteq I$, $|I''| \geq |I|^{1/4}$, and it has one of the following properties: For all $1 \leq i < m$,*

(L1) $r(s_i) < r(s_{i+1})$ and $c_i < c_{i+1}$ (Figure 2(a)),

(L2) $r(s_i) < r(s_{i+1})$ and $c_i > c_{i+1}$ (Figure 2(b)),

(L3) $r(s_i) > r(s_{i+1})$ and $c_i < c_{i+1}$ (Figure 2(c)),

(L4) $r(s_i) > r(s_{i+1})$ and $c_i > c_{i+1}$ (Figure 2(d)).

PROOF. By Dilworth's theorem, there exists a subsequence $I' \subseteq I$ of length at least $\sqrt{|I|}$ such that the x-coordinates of the right endpoints are either monotonically increasing or monotonically decreasing. Again applying Dilworth's theorem to I', one can find a subsequence $I'' \subseteq I'$ of length at least $\sqrt{|I'|} \geq |I|^{1/4}$ such that c_i for $s_i \in I''$ are either monotonically increasing or monotonically decreasing.

If I' is increasing and I'' increasing (resp. decreasing), then the sequence is of type $L1$ (resp. $L2$). If I' is decreasing and I'' increasing (resp. decreasing), then the sequence is of type $L3$ (resp. $L4$). □

We refer to a subsequence of pairwise-disjoint segments of S that satisfies one of $(L1)$–$(L4)$ property as an *l-monotone* sequence. The following property of l-monotone sequences allows us to compute them efficiently.

Lemma 2.5 *Let $S' = \langle s_1, \ldots, s_m \rangle$ be a sequence of segments so that one of the conditions $(L1)$–$(L4)$ is satisfied and $s_i \cap s_{i+1} = \emptyset$ for all i. Then S' is an l-monotone sequence.*

PROOF. It is clear that for segments of type $(L1)$–$(L4)$, two segments s_i and s_j, for $i < j-1$, cannot intersect without either s_i intersecting s_{i+1} or s_j intersecting s_{j-1}. Therefore if $s_i \cap s_{i+1} = \emptyset$ for all i, then the segments are pairwise non-intersecting and hence l-monotone. □

By Lemma 2.5, the segments in any sequence satisfying one of $(L1)$–$(L4)$ are pairwise non-intersecting if the adjacent segments do not intersect (See Figure 2).

We present an algorithm that, given a sequence S of segments, computes the longest l-monotone subsequence of each type. By Lemma 2.4, the longest of them is an independent set of size at least $(\alpha(S))^{1/4}$. We describe an algorithm for computing the longest l-monotone subsequence of type $(L1)$. The others can be computed analogously.

Define S_j to be the set of segments such that $s_k \in S_j$ if

(i) $k < j$, (ii) $r(s_k) < r(s_j)$, (iii) $c_k < c_j$, (iv) $s_k \cap s_j = \emptyset$.

Let $\Phi(j)$ be the longest l-monotone subsequence of S_j of type $(L1)$ that contains s_j. Set $\phi(j) = |\Phi(j)|$. We wish to compute $\max_{1 \leq j \leq n} \phi(j)$.

Lemma 2.6 *For $1 \leq j \leq n$,*

$$\phi(j) = \max_{s_k \in S_j} \phi(k) + 1.$$

PROOF. Let $\Phi(j) = \langle s_{a_1}, \ldots, s_{a_u = j} \rangle$. Clearly $\langle s_{a_1}, \ldots, s_{a_{u-1}} \rangle \subseteq S_{a_{u-1}}$ is an l-monotone sequence, and $s_{a_{u-1}} \in S_j$. Therefore $\phi(a_{u-1}) \geq u - 1$ and $\phi(j) \leq \phi(k) + 1$ for all $s_k \in S_j$. Conversely, Lemma 2.5 implies that for any $s_k \in S_j$, the sequence $\langle \Phi(k) \circ s_j \rangle$ is an l-monotone sequence. Hence, $\phi(j) \geq \phi(k) + 1$. □

A naive approach takes $O(n)$ time to compute each $\phi(j)$, provided that $\phi(k)$, for all $k < j$, have already been computed. This yields an $O(n^2)$ time algorithm. We now describe a more sophisticated approach by exploiting geometry to compute each $\phi(j)$ in $O(n^{1/3}\log^c n)$ amortized time.

Let $M_j = \langle s_1, \ldots, s_j \rangle$. We compute $\phi(j)$ sequentially for $j = 1 \ldots n$, maintaining a data structure Ψ that stores all the segments of M_j. Given the segment s_{j+1}, the data structure returns $\max_{s_k \in S_{j+1}} \phi(k)$. Once we have computed $\phi(j)$, we insert the segment s_j together with its weight $\phi(j)$ into the data structure. Note that after we have inserted $\phi(j)$, the data structure stores all the segments in the set M_{j+1}.

We now describe Ψ, a three-level data structure, that stores a set E of weighted segments. For a query segment γ intersecting the y-axis, it returns the maximum weight of a segment s in E so that $r(s) \leq r(\gamma), \gamma \cap s = \emptyset$, and γ intersects the y-axis above s. The first-level is a balanced binary search tree $T_r(S)$ on the x-coordinates of the right endpoints of the segments in S. Let $\mathcal{C}_u$ denote the "canonical" subset of segments stored in the subtree rooted at $u \in T_r(S)$. For each node u, the second-level data structure is a balanced binary search tree $T_c(\mathcal{C}_u)$ on $\{c(s) \mid s \in \mathcal{C}_u\}$. Let $\mathcal{C}_v^u \subseteq \mathcal{C}_u$ denote the set of segments stored in the subtree rooted at $v \in T_c(\mathcal{C}_u)$. Finally, for the set of segments $\mathcal{C}_v^u$, we construct a segment-intersection data structure $\mathcal{D}(\mathcal{C}_v^u)$ as described in [1]. It stores a family of canonical subsets of $\mathcal{C}_v$ in a tree-like structure. The total size of the data structure is $O(n^{4/3}\log^c n)$, and it can be constructed in $O(n^{4/3}\log^c n)$ time. For a query segment γ, we can report in $O(n^{1/3}\log^c n)$ time the segments of $\mathcal{C}_v^u$ not intersecting γ as a union of $O(n^{1/3}\log^c n)$ canonical subsets. For each canonical subset $A \subseteq \mathcal{C}_v^u$, we store the maximum weight w_A of a segment in A. The overall data structure Ψ can be constructed in time $O(n^{4/3}\log^c n)$.

For a query segment γ, we wish to report $\max \phi(s)$, where the maximum is taken over all segments s of E so that $r(s) \leq r(\gamma)$, $s \cap \gamma = \emptyset$, and the intersection point of s with the y-axis lies below that of γ. We query the first-level tree T_r with the right endpoint of γ and identify a set V_1 of $O(\log n)$ nodes so that $\bigcup_{u \in V_1} \mathcal{C}_u$ is the set of segments whose right endpoints lie to the left of $r(\gamma)$. Next, for each $u \in V_1$, we compute a set $V_2(u)$ of $O(\log n)$ nodes s.t. $\bigcup_{v \in V_2(u)} \mathcal{C}_v^u$ is the set of segments of $\mathcal{C}_u$ that intersect the y-axis below γ does. For each $v \in V_2(u)$, we compute the maximum weight w_v of a segment in $\mathcal{C}_v^u$ that does not intersect γ, using the third-level data structure. We return $\max_{u \in V_1} \max_{v \in V_2(u)} w_v$. The total time spent is $O(n^{1/3}\log^c n)$.

Finally, we can use the standard dynamization techniques by Bentley and Saxe [4] to handle insertions in the data structure. Since the data structure can be constructed in $O(n^{4/3}\log^c n)$ time, the amortized insertion time is $O(n^{1/3}\log^{c+1} n)$. However, in our applications, we know in advance all the segments that we want to insert – they are the segments of S. So we can somewhat simplify the data structure as follows. Set the weight of all segments to 0, and construct Ψ on all the segments of S. When we wish to insert a segment, we update its weight and update the weight of appropriate canonical subsets at the third level of Ψ. Omitting all the details, we conclude the following.

Theorem 2.7 *Given a set S of n segments in $\mathbb{R}^2$, all intersecting a common vertical line, one can compute an independent set of size at least $\alpha(S)^{1/4}$ in time $O(n^{4/3}\log^c n)$.*

2.3 Independent Set for Arbitrary Segments

Let S be a set of arbitrary segments in $\mathbb{R}^2$. We describe a recursive algorithm for computing an independent set of S. Let l be the vertical line passing through the median x-coordinate of the right endpoints of segments in S, i.e., at most $\lceil n/2 \rceil$ segments have their right endpoints on each side of l. We partition S into three sets, S_L, S_R and S^*. S_L (resp. S_R) is the subset of segments that lie completely to the left (resp. right) of l, and S^* is the subset of segments whose interiors intersect l.

We compute an independent set I^* of S^*, and recursively compute an independent set I_L (resp. I_R) of S_L (resp. S_R). Since the segments in S_L do not intersect any segments in S_R, $I_L \cup I_R$ is an independent set of $S_L \cup S_R$. We return either I^* or $I_L \cup I_R$, whichever is larger.

Suppose our algorithm computes an independent set of size at least $\mu(n, \alpha)$, where $\alpha = \alpha(S)$. Let $n_L = |S_L|, n_R = |S_R|, \alpha_L = \alpha(S_L), \alpha_R = \alpha(S_R)$ and $\alpha^* = \alpha(S^*)$. Suppose the algorithm for computing an independent set of S^* returns a set of size at least $\zeta(\alpha^*)$. Then

$$\mu(n, \alpha) \geq \max\{\mu(n_L, \alpha_L) + \mu(n_R, \alpha_R), \zeta(\alpha^*)\},$$

where $\alpha_L + \alpha_R + \alpha^* \geq \alpha$, $n_L, n_R \leq n/2$, $n_L \geq \alpha_L$, and $n_R \geq \alpha_R$. Since μ and ζ are sub-linear functions, it can be argued that

$$\mu(n, \alpha) \geq \max\{\mu(n/2, \alpha - \alpha^*), \zeta(\alpha^*)\}. \tag{2}$$

We now show that the solution to the above recurrence is

$$\mu(n, \alpha) \geq \zeta\left(\frac{\alpha}{2\log(2n/\alpha)}\right).$$

Expanding (2), we obtain

$$\begin{aligned}
\mu(n, \alpha) \geq \quad & \max\{\mu(n/2, \alpha - \alpha_1), \zeta(\alpha_1)\} \\
\geq \quad & \max\{\mu(n/2^2, \alpha - \alpha_1 - \alpha_2), \zeta(\alpha_1), \zeta(\alpha_2)\} \\
& \vdots \\
\geq \quad & \max\{\mu(n/2^j, \alpha - \sum_{i=1}^{j} \alpha_i), \zeta(\alpha_1), \ldots, \zeta(\alpha_j)\} \\
\geq \quad & \max\{\zeta(\alpha_1), \ldots, \zeta(\alpha_l)\}
\end{aligned}$$

where l is the depth of recursion and $\sum_{i=1}^{l} \alpha_i = \alpha$. Moreover, for all j,

$$\frac{n}{2^j} \geq \alpha - \sum_{i=1}^{j} \alpha_i. \tag{3}$$

We claim that

$$\max_{1 \leq i \leq l} \alpha_i \geq \frac{\alpha}{2\log(2n/\alpha)}.$$

Indeed if it were not true, then $l \geq 2\log(2n/\alpha)$. Then for $j = \log(2n/\alpha)$, $n/2^j = \alpha/2$, while $\alpha - \sum_{i=1}^{j} \alpha_i > \alpha - \frac{j\alpha}{2\log(2n/\alpha)} = \alpha/2$, thereby contradicting (3). The proof then follows.

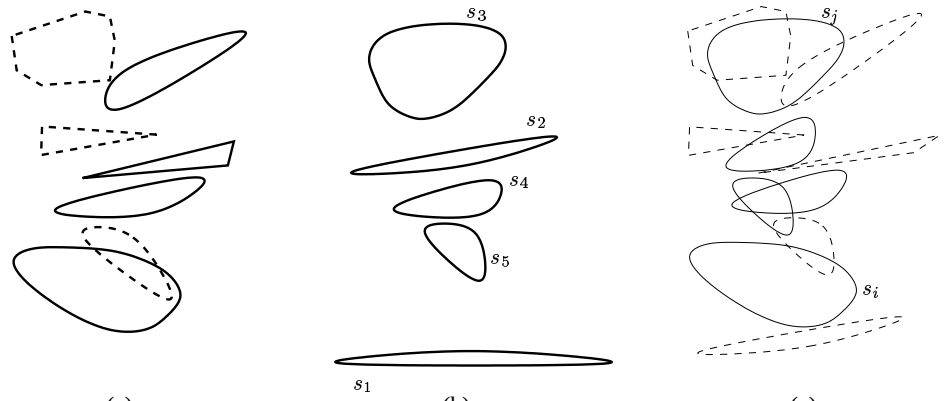


Figure 3: (a) Sequences of type $(C1)$ (bold) and $(C2)$ (dashed) (b) Sequence of type $(C3)$, (c) S_{ij} (solid)

If we can compute the independent set of S^* in time $t(n)$, then the running time of the algorithm is $O(t(n)\log n)$. If $t(n) \geq n^{1+\epsilon}$, then the running time is $O(t(n))$. Hence we conclude the following.

Theorem 2.8 *Let A be a set of m segments, all of which intersect a common vertical line. Suppose we can compute an independent set of A of size at least $\zeta(\alpha(A))$ in time $t(m)$. Then for any set S of n segments in $\mathbb{R}^2$, we can compute an independent set of size at least $\zeta(\alpha/2\log(2n/\alpha))$ where $\alpha = \alpha(S)$. The running time is $O(t(n))$ if $t(n) \geq n^{1+\epsilon}$, and $O(t(n)\log n)$ otherwise.*

Corollary 2.1 *For a set S of n segments in $\mathbb{R}^2$, one can compute an independent set of size at least (i) $(\alpha/2\log(2n/\alpha))^{1/2}$ in time $O(n^3)$ and (ii) $(\alpha/2\log(2n/\alpha))^{1/4}$ in time $O(n^{4/3}\log^c n)$.*

3 Independent Set for Convex Objects

We now describe how the results of the previous section can be extended to find an independent set in a set of convex objects in $\mathbb{R}^2$. Let $S = \{s_1, \ldots, s_n\}$ be a set of convex objects in $\mathbb{R}^2$, and let $\mathcal{I}^*(S)$ be a maximum independent set of S. As for segments, we describe an algorithm for the case in which all objects in S intersect the y-axis. We can then use the approach in Section 2.3 to handle the general case. Define $l(s_i)$ (resp. $r(s_i)$) to be the smallest (largest) x-coordinate of all the points $p \in s_i$. Let c_i be the maximum y-coordinate of the intersection of s_i with the y-axis. Again assume that S is sorted in increasing order of the x-coordinates of the leftmost endpoints. An application of Dilworth's theorem similar to Lemma 2.4 gives the following.

Lemma 3.1 *Given a set $I \subseteq S$ of pairwise-disjoint convex objects, there exists a subsequence $I' = \langle s_1, \ldots, s_m \rangle$, where $|I'| \geq |I|^{1/3}$ and I' has* one *of the following structure: For all $1 \leq i < m$*

(C1) $r(s_i) < r(s_{i+1})$ and $c_i < c_{i+1}$ (Figure 3(a)), or

(C2) $r(s_i) < r(s_{i+1})$ *and* $c_i > c_{i+1}$ *(Figure 3(a)), or*

(C3) $r(s_i) > r(s_{i+1})$ *(Figure 3(b)).*

Sequences satisfying condition $(C1)$ or $(C2)$ can be computed using a dynamic programming approach similar to the one in Section 2.2. We outline the algorithm for computing a longest subsequence of type $(C3)$.

For $1 \leq i < j \leq n$ such that $s_i \cap s_j = \emptyset$, let $S_{ij} \subseteq S$ denote the subsequence of segments s_k so that

(i) $c_i < c_k < c_j$,

(ii) $\max\{l(s_i), l(s_j)\} < l(s_k) < r(s_k) < \min\{r(s_i), r(s_j)\}$, and

(iii) $s_i \cap s_k = \emptyset$ and $s_j \cap s_k = \emptyset$.

See Figure 3(c). Let $\Phi(i,j) \subseteq S_{ij}$ denote the longest subsequence of type $(C3)$ of S_{ij}. Set $\phi(i,j) = |\Phi(i,j)|$. Then we can prove the following.

Lemma 3.2 *For all* $1 \leq i < j \leq n$,

$$\phi(i,j) = \max_{s_k \in S_{ij}} \phi(i,k) + \phi(k,j) + 1. \tag{4}$$

We can compute $\phi(i,j)$ using a dynamic-programming approach. Assuming $\phi(i,k)$ and $\phi(k,j)$ have already been computed, we only need to compute the set S_{ij}. Suppose we can preprocess S in time $\tau(S)$ so that we have the following information at our disposal: $(P1)$ $l(s_i), r(s_i), c_i$ for each $s_i \in S$, $(P2)$ whether $s_i \cap s_j = \emptyset$ for each $s_i, s_j \in S$. Then the set S_{ij} can be computed in time $O(n)$. Hence, we can compute $\phi(1,n)$ in $O(n^3 + \tau(S))$ time. Plugging this procedure into the recursive scheme of Section 2.3 we obtain the following.

Theorem 3.3 *Let* S *be a set of* n *convex objects in* $\mathbb{R}^2$ *so that* $(P1)$*-*$(P2)$ *can be computed in* $\tau(S)$ *time. Then an independent set of size at least* $(\alpha/2\log(2n/\alpha))^{1/3}$ *can be computed in time* $O(n^3 + \tau(S))$.

4 Independent Set for Axis-Parallel Rectangles

Let $S = \{s_1, \ldots, s_n\}$ be a set of n rectangles in $\mathbb{R}^2$, and let $\omega(S)$ denote the size of the largest clique in the intersection graph of S. Without loss of generality, we can assume that no rectangle completely contains any another rectangle. We first consider two special cases of rectangle intersection graphs — the *piercing* and *non-piercing* intersection subgraphs, derived by defining the following partial order among the rectangles.

Given two rectangles s_1 and s_2, $s_1 \prec s_2$ if and only if s_1 intersects both vertical edges of s_2, and s_2 intersects both horizontal edges of s_1 (See Figure 4(a)). Clearly, if $s_1 \prec s_2$, then s_1 and s_2 intersect, and neither contains any vertex of the other. Note that $\prec$ is a transitive relation: $s_1 \prec s_2 \prec s_3$ implies $s_1 \prec s_3$. If $s_1 \prec s_2$, we say that s_1 and s_2 *pierce*, and that s_2 is pierced by s_1.

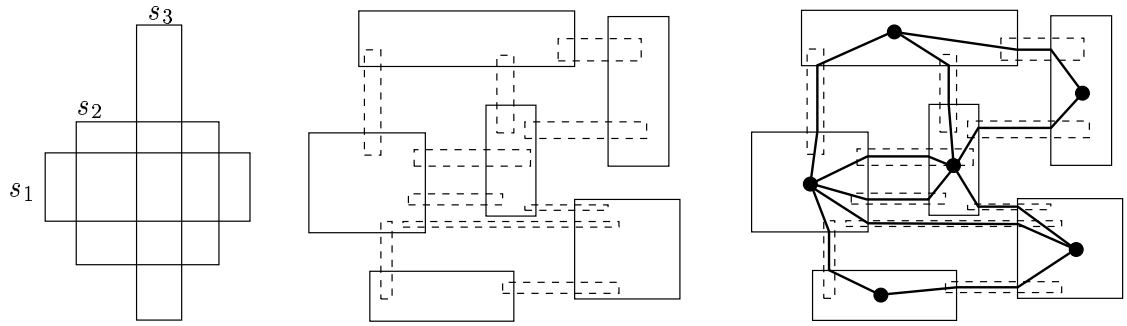


Figure 4: (a) $s_1 \prec s_2 \prec s_3$, (b) The set S' (solid) and the set I' (dashed), (c) Mapping rectangles in S' to vertices, and rectangles in I' to edges.

4.1 Piercing and Non-Piercing Rectangle Intersection Graphs

Given a set S, the relation $\prec$ partitions the edges in the intersection graph G_S into two sets E_1 and E_2: given an edge $\{a, b\} \in E(G_S)$, $\{a, b\} \in E_1$ if $a \prec b$ or $b \prec a$, and $\{a, b\} \in E_2$ otherwise.

Piercing Intersection Graphs. Define the piercing intersection graph of S as the directed graph $G_p = (V, E)$, i.e., each vertex in $V(G_p)$ corresponds to a rectangle in S, and there is a directed edge (a, b) between two vertices a and b if b is pierced by a, i.e., $(a, b) \in E(G_p)$ if and only if $a \prec b$.

Lemma 4.1 *Given a set S of n rectangles in $\mathbb{R}^2$, let G_p be the piercing graph of S. Then $\omega(G_p) \cdot \alpha(G_p) \geq n$, and the optimal independent set in G_p can be computed in polynomial time.*

PROOF. It follows from the transitivity of $\prec$ that G_p is a transitive graph. It is a well-known fact that all transitive graphs are *perfect* graphs, i.e., for every induced subgraph of G, the size of the maximum clique equals the chromatic number. Therefore the vertices of G_p can be partitioned into $\omega(G_p)$ subsets, each of which is an independent set in G_p. Since the largest subset has at least $n/\omega(G_p)$ vertices, we conclude that $\omega(G_p) \cdot \alpha(G_p) \geq n$.

By a classical result of Grötschel *et al.* [13], a largest independent set of a perfect graph can be computed in polynomial time. □

Non-Piercing Intersection Graphs. We now consider the case of pairwise non-piercing rectangles. Let S be a set of n rectangles such that no two rectangles pierce (although they could intersect); the intersection graph of S is called the non-piercing intersection graph and denoted as G_{np}. First, note that computing the optimal independent set of S remains NP-hard since the construction in [22] proving the NP-hardness for intersection graphs of rectangles uses only non-piercing rectangles.

We call a subset $S' = \{s_{i_1}, \ldots, s_{i_m}\} \subseteq S$ *r-maximal* if it satisfies the following four conditions:

(A1) S' is an independent set,

(A2) Every $s \in S \setminus S'$ intersects some $s' \in S'$,

(A3) For each $s' \in S'$, there is at most one rectangle $s \in \mathcal{I}^*(S)$ such that s intersects s' and does not intersect any other rectangle in S', and

(A4) For every pair of disjoint rectangles $s', t' \in S'$, there are at most two rectangles $s, t \in S \setminus S'$ such that $s \cap t = \emptyset$, and s, t intersect both s' and t', and no other rectangle in S'.

Lemma 4.2 *Let $S' = \{s_{i_1}, \ldots, s_{i_m}\} \subseteq S$ be an r-maximal set. Then $|S'| \geq \alpha(S)/c_0$, where $c_0 \leq 11$.*

PROOF. Let $I = \mathcal{I}^*(S)$ be the set of rectangles in the maximum independent set. We will charge each rectangle in I to a rectangle in S' such that each rectangle in S' is charged at most a constant c_0 times. The lemma then follows.

Let s be a rectangle in I. By maximality condition $(A2)$, s intersects at least one rectangle in S', and since the rectangles in S are pairwise non-piercing, s does not pierce any rectangle in S'. We charge s to a rectangle in S' as follows. First, if $s \in S'$, we charge s to itself. Clearly, each rectangle in $S' \cap I$ receives only one unit of charge. Otherwise, we charge s as follows:

C1. s only intersects one rectangle $s' \in S'$. Charge s to s'. By condition $(A3)$, each rectangle in S' receives at most one unit of charge.

C2. s intersects s' in a corner (i.e., s contains a vertex of s'). Charge s to s'. Since I and S' are independent sets, each rectangle in S' receives at most four units of charge.

Let $I' \subseteq I$ be the set of rectangles of I that have *not* yet been charged. Thus far each rectangle in S' has received at most 5 charges, so therefore $|I \setminus I'| \leq 5|S'|$. We now bound the number of rectangles in I'. I' has the following property: each rectangle in I' intersects exactly two rectangles in S' (see Figure 4(b)); C1 and C2 above deal with all other intersection possibilities. We map each rectangle in S' and I' to a vertex and an edge, respectively, by constructing a bipartite multi-graph $G_{I'} = (V, E)$ where each vertex in V corresponds to a rectangle in S', and there is an edge between v_j and v_k if there is a rectangle $s \in I'$ that intersects both s_{i_j} and s_{i_k}. Clearly each rectangle in I' maps to an edge in $G_{I'}$.

We now show that $G_{I'}$ is a planar multi-graph as follows. Replace each rectangle $s' \in S'$ with its center $c(s')$, and each rectangle $s \in I'$ with a polygonal curve (consisting of three piecewise-linear segments) as illustrated in Figure 4(c). It is clear, given the independence property of the rectangles in I' and S', that the above embedding is non-intersecting. Hence $G_{I'}$ is a planar multi-graph.

For a planar graph, the number of edges is at most thrice the number of vertices. Since $G_{I'}$ is a planar multi-graph with at most two edges between any pair of vertices by condition (A4), we have $|I'| = |E| \leq 6|V| = 6|S'|$. Hence, combining the bounds,

$$|I| = |I'| + |I \setminus I'| \leq 6|S'| + 5|S'| = 11|S'|.$$

□

Lemma 4.3 *Given a set S of n rectangles in $\mathbb{R}^2$ such that no two rectangles pierce, let G_{np} denote the corresponding (non-piercing) intersection graph. Then $\omega(G_{np}) \cdot \alpha(G_{np}) \geq \frac{n}{4}$, and a c_0-approximation to the maximum independent set can be computed in polynomial time,*

PROOF. First, by Turán's Theorem [19], there exists an independent set of size at least $n^2/|E|$. Second, for each edge $e = \{s_i, s_j\}$, due to the properties of non-piercing graphs, either s_i contains a vertex of s_j or vice versa. Let c_i denote the number of times rectangle s_i's vertex is contained in another rectangle. If s_i contains a vertex of s_j, charge the edge e to vertex s_j by incrementing c_j, otherwise increment c_i. Clearly, $\sum_{i=1}^{n} c_i = |E|$, and hence there exists a rectangle s_k whose vertices are contained in more than $|E|/n$ rectangles, and therefore at least one vertex contained in more than $|E|/4n$ rectangles. These rectangles share a common point, and thus form a clique of size greater than $|E|/4n$. Hence,

$$\omega(G_{np}) \cdot \alpha(G_{np}) \geq \frac{n^2}{|E|} \cdot \frac{|E|}{4n} = \frac{n}{4}.$$

We describe an iterative algorithm to compute a r-maximal set, which, by Lemma 4.2, is a constant-factor approximation of the maximum independent set of G_S. In the beginning of the i-th iteration the algorithm maintains a set S_i of i pairwise-disjoint segments. In the i-th iteration, the algorithm checks whether conditions (A2)–(A4) are satisfied. If the answer is yes, we return S_i, as it is a r-maximal set. Otherwise,

1. Condition (A2) violated. Then there exists a set $s \in S \setminus S'$ that does not intersect any $s' \in S'$. Set $S_{i+1} = S_i \cup \{s\}$.

2. Condition (A3) violated. Then there exist two disjoint rectangles $s, t \in S \setminus S'$ that intersect some $s' \in S'$ but no other rectangle in S'. Set $S_{i+1} = S_i \setminus \{s'\} \cup \{s, t\}$.

3. Condition (A4) violated. Then there exist three pairwise-disjoint rectangles $s, t, u \in S \setminus S'$ that all intersect $s', t' \in S'$, but no other rectangle in S'. Set $S_{i+1} = S_i \setminus \{s', t'\} \cup \{s, t, u\}$.

It is clear that S_i is a set of pairwise-disjoint segments. The process can continue for at most n iterations, and the final set S_j is obviously a r-maximal set. Since each of the conditions (A2)–(A4) can be checked in polynomial time, the total running time is polynomial. □

4.2 General Rectangle Intersection Graphs

Combining Lemma 4.1 and Lemma 4.3, we attain the following.

Lemma 4.4 *Given a set S of n rectangles in $\mathbb{R}^2$, an independent set of size at least $\frac{\alpha(S)}{d_0 \cdot \omega(S)}$, for some constant $d_0 \leq 4c_0 \leq 44$, can be computed in polynomial time.*

PROOF. By Lemma 4.1, compute the maximum independent set, say $A \subseteq S$, in the piercing graph of S. Note the following:

(i) $|A| \geq \alpha(S)$ as an independent set in S is an independent set in piercing graph of S, and

(ii) $\omega(A) \leq \omega(S)$, since a clique in A is a clique in S.

By Lemma 4.3, $\alpha(A) \cdot \omega(A) \geq |A|/4$. Hence, using (i) and (ii) above, it follows that $\alpha(A) \geq |A|/4\omega(A) \geq \alpha(S)/4\omega(S)$. Since the intersection graph of A is non-piercing, Lemma 4.3 gives an algorithm which returns an independent set of size at least $\alpha(A)/c_0 \geq \alpha(S)/4c_0\omega(S)$. □

From now on, let $\alpha(S) = \beta n$, for some $\beta \leq 1$.

Theorem 4.5 *Given S with $\alpha(S) = \beta n$, one can compute an independent set of size $\frac{\alpha(S)}{4d_0 \cdot (1/\beta)}$ in polynomial time.*

PROOF. The algorithm repeatedly extracts large cliques (one can compute the maximum clique in rectangle intersection graphs in polynomial time [16]) until a good independent set is found, as follows. Set $S_0 = S$, and let S_i be the set of rectangles in the i-th iteration. For now assume that the value of β is known. At the i-th iteration, if there exists a clique $\mathcal{C}$ of size at least $2/\beta$, remove $\mathcal{C}$ from S_i, i.e., set $S_{i+1} = S_i \setminus \mathcal{C}$, and reiterate. If no such clique exists, compute the maximum independent set, say A, in the piercing graph of S_i, and return the maximum independent set in A. Assume the algorithm stops after j iterations, i.e. $\omega(S_j) \leq 2/\beta$. Note that $j \leq n/(2/\beta) = \beta n/2 = k/2$. Since at most one rectangle from the independent set can be in a clique, each iteration removes at most one rectangle from the optimal independent set of S, hence $\alpha(S_j) \geq k - j$. From Lemma 4.4, one can thus compute an independent set of size at least

$$\frac{\alpha(S_j)}{d_0\omega(S_j)} \geq \frac{k-j}{d_0(2/\beta)} \geq \frac{k - k/2}{d_0(2/\beta)} = \frac{k}{(4d_0/\beta)},$$

yielding the desired result. Note that we do not know the value of β, but can run the above algorithm for all the n possible values, and return the maximum. □

5 Conclusions

In this paper we have presented algorithms for approximating the maximum independent set in the intersection graphs of convex objects in the plane. The approximation ratio is better if the convex objects are line segments.

All the algorithms described in this paper have the same overall approach. They first prove the existence of a large independent subset with some special (separator-like) properties. They then show that this subset can be computed *exactly* from among the entire set (we used dynamic-programming). One approach toward improving these results is to show the existence of independent subsets of larger size, which are still computable in polynomial time.

We leave it as an open problem whether the approximation ratios can be improved. In particular, is it possible to design a $\sqrt{\alpha}$-approximation algorithm for the case of general convex objects (all intersecting a vertical line)? Similarly, is it possible to approximate the independent set of line segments better than $\sqrt{\alpha}$. For axis-parallel rectangles, devising an algorithm with approximation ratio $o(\log n)$ remains an intriguing open problem. We have made a step toward crossing the logarithmic bound in this paper, but a general-case constant-factor algorithm remains elusive.

References

[1] P. K. Agarwal and J. Erickson, Geometric range searching and its relatives, *Advances in Discrete and Computational Geometry* (B. Chazelle, J. E. Goodman, and R. Pollack, eds.), American Mathematical Society, 1999, pp. 1–56.

[2] P. K. Agarwal, M. van Kreveld, and S. Suri, Label placement by maximum independent set in rectangles, *Computational Geometry: Theory and Applications*, 11 (1998), 209–218.

[3] B. S. Baker, Approximation algorithms for NP-complete problems on planar graphs, *J. ACM*, 41 (1994), 153–180.

[4] J. L. Bentley and J. B. Saxe, Decomposable searching problems I: Static-to-dynamic transformation, *J. Algorithms*, 1 (1980), 301–358.

[5] P. Berman, B. DasGupta, S. Muthukrishnan, and S. Ramaswami, Efficient approximation algorithms for tiling and packing problems with rectangles, *Journal of Algorithms*, 41 (2001), 443–470.

[6] P. Berman and T. Fujito, Approximating independent sets in degree 3 graphs, *Proc. 4th Workshop on Algorithms and Data Structures*, 1995, pp. 449–460.

[7] R. Boppana and M. M. Halldórsson, Approximating maximum independent sets by excluding subgraphs, *Proc. 2nd Scandinavian Workshop on Algorithm Theory*, 1990, pp. 13–25.

[8] T. M. Chan, Polynomial-time approximation schemes for packing and piercing fat objects, *Journal of Algorithms*, 46 (2003), 178–189.

[9] T. M. Chan, A note on maximum independent sets in rectangle intersection graphs, *Information Processing Letters*, 89 (2004), 19–23.

[10] P. Dilworth, A decomposition thorem for partially ordered sets, *Annals of Mathematics*, 51 (1950), 161–166.

[11] T. Erlebach, K. Jansen, and E. Seidel, Polynomial-time approximation schemes for geometric graphs, *Proc. of the 12th Annual ACM-SIAM Sympos. on Discrete Algorithms*, 2001, pp. 671–679.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, 1979.

[13] M. Grotschel, L. Lovász, and A. Schrijver, Polynomial algorithms for perfect graphs, *Ann. Discrete Math.*, 21 (1981), 211–356.

[14] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, *Acta Math.*, 182 (1999), 105–142.

[15] H. B. Hunt, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns, NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs, *J. Algorithms*, 26 (1998), 238–274.

[16] H. Imai and T. Asano, Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane, *J. Algorithms*, 4 (1983), 310–323.

[17] J. Kratochvil and J. Matoušek, Intersection graphs of segments, *Journal of Combinatorial Theory, Series B*, 62 (1994), 289–315.

[18] E. Malesinska, Graph-theoretical models for frequency assignment problems, Ph.D. Thesis, Technische Universitit Berlin, 1997.

[19] J. Pach and P. K. Agarwal, *Combinatorial Geometry*, John Wiley & Sons, New York, NY, 1995.

[20] J. Pach and G. Tardos, Cutting glass, *Proc. 16th Annual Sympos. on Computational Geometry*, 2000, pp. 360–369.

[21] C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *J. Comput. System Sci.*, 43 (1991), 425–440.

[22] C. S. Rim and K. Nakajima, On rectangle intersection and overlap graphs, *IEEE Trans. on Circuits and Systems*, 42 (1995), 549–553.