Polynomial Time Approximation Schemes for the Geometric Independent Set Problem

Abraham Hinteregger
Matr.: 1025914
Curriculum: 066931
`oerpli@outlook.com`

March 4, 2018

Abstract

The geometric independent set problem is a special case of the independent set problem where the graph is the intersection graph of a set of geometric objects. The challenge of the geometric independent set problem is to find the biggest set of non-intersecting objects, i.e. the maximum independent set (MIS). In this report I will present polynomial time approximation schemes (PTAS) for the geometric independent set problem for unit disk graphs (UDG) and for unit height rectangles. The geometric independent set problem is in general $\mathcal{NP}$- hard and therefore intractable. The algorithms presented here use different approaches to guarantee a polynomial running time for a constant approximation quality.

Contents

1 Introduction

The geometric independent set problem, i.e. finding a (maximum) set of non-intersecting geometric shapes is a problem that can arise in many different fields. In this report I will introduce algorithms to give approximate solutions for the MIS of unit disk graphs as well as for the geometric independent set problem for unit height rectangles. Unit disks may correspond to the area of influence of radio towers where frequencies can only be assigned to radio towers where this areas do not intersect (i.e. an independent set) to prevent interference [5]. The maximum independent set of unit height rectangles could correspond to a labelling of a two dimensional map with a uniform font [1].

1.1 Maximum independent set problem for graphs

An independent set of a graph is a set of vertices that are not adjacent (connected with an edge). Finding a maximum independent set of an arbitrary graph is known to be an $\mathcal{NP}$-complete problem [4]. The best known exact algorithms have a runtime $\mathcal{O}(1.22^n)$ [3, 6] (already quite good compared to the $\mathcal{O}(n^2 2^n)$ naive brute force approach resulting from checking every subset of vertices).

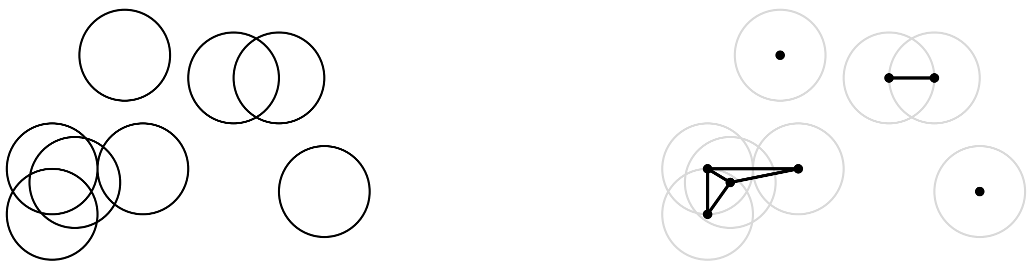
If the graph is an interval intersection graph (see Section 1.3) the MIS can be found in polynomial time (similar to the algorithm described in Section 3.1.1) though for many (most) other intersection graphs it is still $\mathcal{NP}$-complete but approximate solutions can be found efficiently.

1.2 Polynomial time approximation scheme

As finding exact solutions for $\mathcal{NP}$-hard problems is not feasible in most cases, algorithms that solve a given instance of an intractable problem in polynomial time with a bounded loss in solution quality are of interest. A so called polynomial time approximation scheme (PTAS) is a family of algorithms that find a solution with a solution quality that is bounded by a factor of $(1 + \varepsilon)$ for any $\varepsilon > 0$.

That is, if the optimal solution is S_{OPT}, the algorithm finds a solution with a quality of at least ${}^{S_{\text{OPT}}}\!/_{1+\varepsilon}$ for a maximization problem and a solution with a quality of at most $(1 + \varepsilon)S_{\text{OPT}}$ for a minimization problem. In both cases the algorithm has runtime polynomial in input-size though the dependency on ε is usually not polynomial.

The subset of problems in $\mathcal{NP}$ (actually $\mathcal{NPO}$ – $\mathcal{NP}$- Optimization) with a polynomial time approximation scheme with constant quality bound (with $\mathcal{O}(1)$-approximation) is also called $\mathcal{APX}$. If the problem admits $1 \pm \varepsilon$- approximation schemes (for arbitrarily small $\varepsilon > 0$) it is in the subset $\mathcal{PTAS}$ of $\mathcal{APX}$.



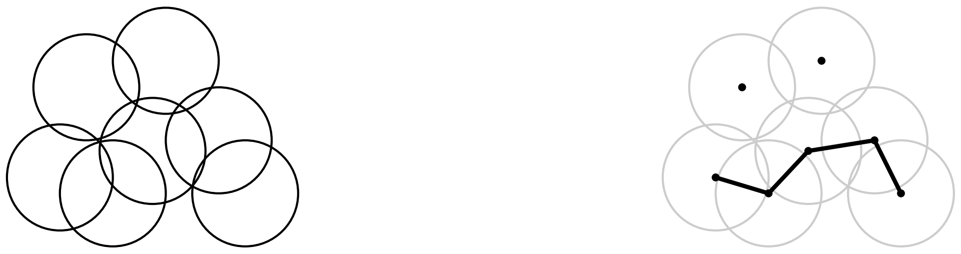
(a) Geometric shapes (unit disks) (b) Corresponding intersection graph

Figure 1: Arrangement of geometric shapes and corresponding intersection graph

1.3 Intersection graphs

An intersection graph of a set of objects represents how these objects intersect each other. Figure 1 shows an example of such an arrangement and its corresponding intersection graph.

It is also possible to reduce the edge set of the intersection graph to e.g. those edges where the corresponding objects intersect on an area bigger than a certain threshold (e.g. finding feasible locations for base stations where overlapping regions are inefficient). An example of this is in Figure 2 with a threshold of 20% of the circle area.



(a) Geometric shapes (unit disks) (b) Corresponding intersection graph

Figure 2: Edges in IG correspond to at least 20% intersecting area of the circles.

1.3.1 Construction of intersection graphs from geometric objects

Given an arrangement $\mathcal{A}$ of n geometric objects its intersection graph $G = (V, E)$ can be obtained in the following way:

- The vertex set V consists of n vertices — each of them uniquely represents one of the geometric shapes of $\mathcal{A}$
- For every pair s_i, s_j of vertices check whether they intersect each other and add the edge $\langle i, j\rangle$ to the edge set E if this is the case.

Obviously the runtime of this scheme is $\mathcal{O}(n^2)$. The worst case performance of finding an intersection graph is always $\mathcal{O}(n^2)$ as there can be a quadratic amount of intersections (if all objects intersect each other). If there are less intersection this can usually be improved to $\mathcal{O}(n \log n + I)$ where I is the amount of intersections (once again, with quadratic upper bound) and the other term is from sorting the objects according to some criterion, e.g. the leftmost point of the object on the x-axis. Sweepline algorithms are used for this, the best known is the Bentley–Ottman algorithm for line segments [2] with runtime $\mathcal{O}(n \log n + nI)$.

1.3.2 Geometric representation of intersection graph

Finding a valid representation, i.e. a mapping $f : V \to \mathbb{R}^d$, from the set of vertices to a vector representation of the geometric objects defined by $x_1, \dots x_d \in \mathbb{R}$ (e.g. for disks in the plane $d = 3$ ($x-, y-$ coordinates and radius), for rectangles $d = 4$ (two points with two coordinates each)) is in many cases a $\mathcal{NP}$-hard problem [7] and it is therefore not feasible to find a geometric representation of a given graph and use the additional information to simplify the general independent set problem.

1.4 Robust algorithms

An algorithm $\mathcal{A}$ computes a function $f : \mathcal{G} \to \mathcal{H}$. If the algorithm is only able to compute the correct result $f(i)$ for $i \in \mathcal{U} \subset G$ there are elements in $\mathcal{G}$ that are not in $\mathcal{U}$ and therefore the algorithm may not compute the correct result (or may not terminate at all).

Definition 1. *An algorithm $\mathcal{A}$ computes f robustly on $\mathcal{U}$ if:*

- *for all instances $i \in \mathcal{U}$ it returns the correct result $f(i)$*
- *for all instances $i \in \mathcal{G} \setminus \mathcal{U}$ the algorithm either returns $f(i)$ or a certificate showing that $i \notin \mathcal{U}$.*

2 M(W)IS for unit disk graphs (UDG)

In this section I will present a robust PTAS for both, the weighted and unweighted unit disk graph independent set problem as introduced by Nieberg et al. [8]. The algorithm does not depend on the geometric representation, though the polynomial runtime is only guaranteed if such a representation exists because it uses the area π of the unit disks to get an upper bound on the amount of disks in an independent set of a certain area.

2.1 Preliminaries

A unit disk graph $G = (V, E)$ is a graph where there exists a geometric representation $f : V \to \mathbb{R}^2$ i.e. a function that maps every vertex to a point to the center point on the real Cartesian plane of the disk in the geometric representation such that:

$$(u, v) \in E \leftrightarrow ||f(u) - f(v)|| \leq 2. \tag{1}$$

As finding this representation is $\mathcal{NP}$-hard it is not feasible to compute a valid representation of the given graph. Furthermore it is also $\mathcal{NP}$-hard to determine if a valid geometric representation even exists [7].

Furthermore, let $\rho = 1 + \varepsilon$ denote the desired approximation ratio where $\varepsilon > 0$.

2.2 MIS for unweighted UDG

The algorithm is given a UDG $G = (V, E)$ and the desired result is a set $I \subseteq V$ that has a cardinality that is at least $\alpha(G)\rho^{-1}$ where $\alpha(G)$ is the maximum size of an independent set in G. The algorithm starts at an arbitrary node $v \in V$ and computes the sets

$$N_r = N_r(v) := \{w \in V | w \text{ has distance at most } r \text{ from } v\}$$

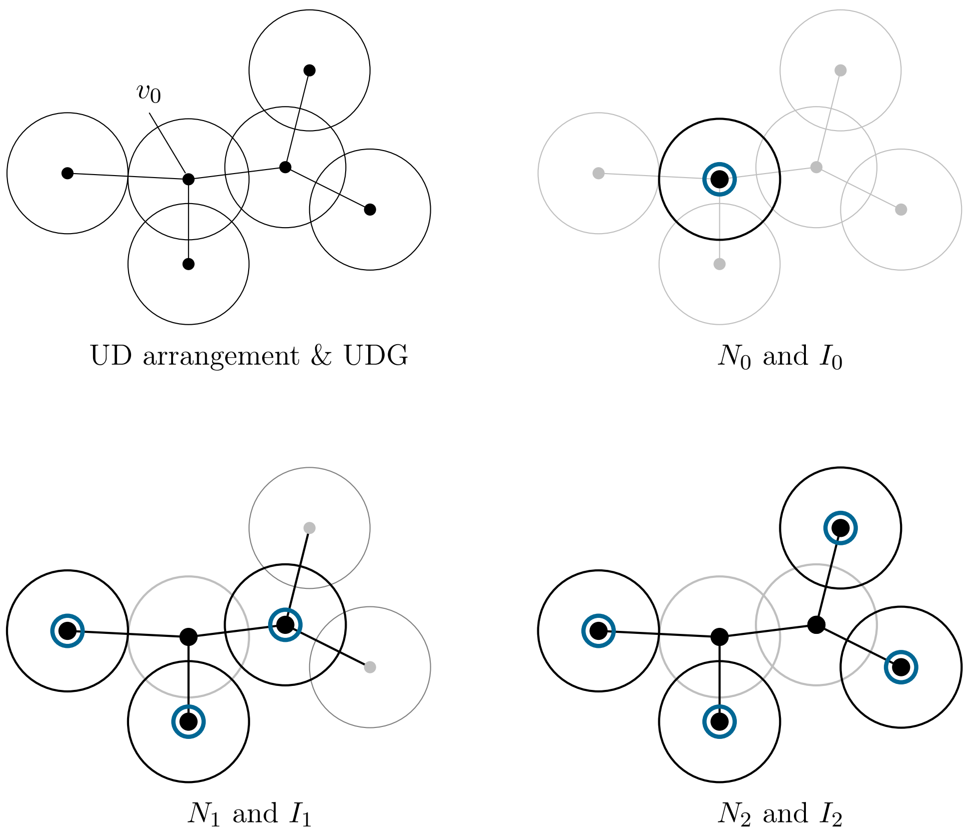


Figure 3: Unit disk graph (with disk representation) and the sets calculated by the algorithm. Nodes in a neighborhood set N_r are marked with ● and nodes in independent sets I_r are marked with ◉. Not yet considered nodes are marked with ● . The algorithm starts at v_0 and takes neighborhoods N_r of increasing size r until the size of the independent set $I_r \subset N_r$ is below a certain threshold.

for $r = 0, 1, \ldots$. Starting from N_0 the algorithm computes the maximum independent set $I_r \subset N_r$ of these r-neighborhoods until the condition

$$|I_{r+1}| > \rho |I_r| \tag{2}$$

is no longer fulfilled - let $\bar{r}$ be the smallest r where equation (2) is violated. Such a $\bar{r}$ must exist and it has a constant upper bound. Figure 3 shows how these neighborhoods are selected on an example graph.

Theorem 1 (Nieberg et al. [8]). *There exists a constant $c = c(\rho)$ such that $\bar{r} \leq c$*

Proof. Due to Equation (1) any vertex $w \in N_r$ satisfies

$$||f(w) - f(v)|| \leq 2r.$$

it is therefore possible to draw a circle with radius $R = 2r + 1$ that contains all disks representing the vertices in N_r. This circle also contains the disk representations of the vertices in I_r which have a disjoint area of π each (as their radius equals 1). This leads to an upper bound on the size of each independent set:

$$|I_r| \leq \pi R^2/\pi = (2r + 1)^2 = O(r^2). \tag{3}$$

From Equation (2)follows:

$$|I_r| > \rho|I_{r-1}| > \ldots > \rho^r|I_0| = \rho^r. \tag{4}$$

Combining (3) and (4) yields

$$\rho^r < |I_r| \leq O(r^2), \tag{5}$$

an inequality (two of them actually) where the lower bound grows asymptotically faster then the upper bound with increasing r – therefore there exists a constant upper bound $c(\rho)$ for r. □

The fact that the size of the independent sets calculated for the various N_r $(r \leq \bar{r})$ implies a polynomial runtime of $\mathcal{O}(n^{C^2})$ (where $C = \mathcal{O}(r) = \mathcal{O}(1/\varepsilon^2 \log(1/\varepsilon))$. The full algorithm proceeds as follows:

1. Calculate the independent set $I_{\bar{r}}$ starting from an arbitrary vertex v.
2. Remove the vertices in $N_{\bar{r}+1}$ from the graph G to get the graph $G' = (V', E') = G \setminus N_{\bar{r}+1}$ (including edges incident to any of the removed vertices).
3. Repeat the previous two steps for G' until there is no vertex left i.e. $G' = (\varnothing, \varnothing)$.
4. Combine all $I_{\bar{r}}$ to get a ρ-approximate maximum independent set.

2.2.1 Proof of correctness & approximation guarantee

The correctness and approximation guarantee follows from the following two theorems by Nieberg et al. [8]:

Theorem 2. *Suppose that we can compute an independent set $I' \subset V \setminus N_{\bar{r}+1}$ of the graph G'. Then $I := I_{\bar{r}} \cup I'$ is an independent set for G.*

Proof. A vertex $v \in I' \subset V'$ has no neighbor $n \in N_{\bar{r}}$ as the distance between v and any such n is at least $\bar{r} + 2$ (else it would have been in $N_{\bar{r}}$ and removed from the vertex set of G'). Therefore I is an independent set. □

Theorem 3. *Suppose inductively that we can compute a ρ-approximate independent set $I' \subset V \setminus N_{\bar{r}+1}$ of the graph G'. Then $I := I_{\bar{r}} \cup I'$ is a ρ-approximate independent set for G.*

Proof. As $\bar{r}$ is chosen in such a way that the maximum independent set of $N_{\bar{r}}$ is a ρ-approximate independent set of $N_{\bar{r}+1}$ (ensured by Equation (2)):

$$|I_{\bar{r}+1}| \leq \rho|I_{\bar{r}}|.$$

Therefore the following holds:

$$\alpha(G[N_{\bar{r}+1}]) \leq \rho|I_{\bar{r}}| = \rho\alpha(N_{\bar{r}}).$$

Furthermore it is obvious that the size of an independent set of a graph is at most as large as the sum of the sizes of independent sets of subgraphs that form a partition of the initial graph. Applying this here yields

$$\begin{aligned}
&\alpha(G) \leq \alpha(G[N_{\bar{r}+1}]) + \alpha(V \setminus G[N_{\bar{r}+1}]) \leq \rho|I| \\
&\alpha(G) \leq \rho|I|,
\end{aligned}$$

and thus proves that the desired approximation guarantee is fulfilled. □

2.3 MWIS for weighted UDG

If the given UDG has a vector $\vec{w}$ assigning positive values w_i to every vertex v_i and the objective is to find an independent set of maximum weight the algorithm described in Section 2.2 can be easily adapted to yield an independent set that has a weight that is at least ρ^{-1} the weight of the independent set with the maximum weight. The following things have to be adapted:

Starting node v_0: Don't start at an arbitrary node v_0 but at the node with the heighest weight.

Stopping criterion: Modify the stopping criterion in Equation (2) to consider the weight instead of only the size. The function $W : V^n \to \mathbb{R}$ maps vertices to their weight and sets of vertices to the sum of their weight.

$$W(I_{r+1}) > \rho W(I_r)$$

Obtaining the bound to get a constant upper bound (as in Equation (5)) is rather straightforward:

Theorem 4 (Nieberg et al. [8]). *There exists a constant $c = c(\rho)$ such that $\bar{r} \leq c$*

Proof. The idea of the proof is the same as for the unweighted case. As the upper bound use:

$$W(I_r) = \sum_{i \in I_r} W(v_i) \leq \sum_{i \in I_r} W(v_0) = |I_r| w_0$$

where w_0 is the weight of v_0 – the vertex with highest weight. And the lower bound use:

$$W(I_r) > \rho W(I_{r-1} > \ldots > \rho^r W(I_0) = \rho^r w_0$$

As it still holds that I_r is bounded by $\mathcal{O}(r^2)$ (using the area of the unit disks) we get the same upper and lower bounds but with an additional constant factor for the weight of the initial node. □

2.4 Robustness

The algorithms outlined in Section 2.2 and Section 2.3 are PTAS with the desired solution quality as long as the graph really is a unit disk graph. To get a robust PTAS for arbitrary graphs from this there are some minor modifications necessary. Observe, that in both cases the approximation quality as well as the overall correctness did not depend on any properties of a UDG. Only for getting the constant bound on the sizes of the independent sets I_r the area of the disks was used.
The only necessary modification is therefore to look whether an independent set of size $|I_r^*| > (2r+1)^2$ can be found and if this is the case return it as certificate that the given graph is not a UDG. Finding this set is also possible in polynomial time as not the maximum independent set $I_r \subset N_r$ has to be found (the maximum size of this independent set would be unbounded and therefore impossible to determine in polynomial time) but only an independent subset with a size of at least $(2r+1)^2 + 1$ is necessary which is still possible in polynomial time.

3 MIS for unit height rectangles

The maximum independent set problem for unit height rectangle is a problem that arises when labelling maps. The goal is to find a maximum non-intersecting set of possible labels in the plane to ensure legibility. In this section I will present a 2-approximation algorithm with $\mathcal{O}(n \log n)$ runtime for the unweighted MIS problem for unit height rectangles introduced by Agarwal et al. [1] which can be improved upon with a dynamic programming approach to get a $(1 + {}^{1}\!/\!{}_{k})$-approximation algorithm with runtime $\mathcal{O}(n \log n + n^{2k-1})$ for $k \geq 1$. The idea of the improvement will be presented here, for the technical details the original paper has to be consulted. For this algorithms the arrangement of rectangles is required.

3.1 A 2-approximation algorithm

The set of n rectangles R with unit height is given. The maximum independent set of rectangles is I_{OPT}. To get a 2-approximate independent set, the algorithm partitions the n rectangles into disjunct subsets, calculates the maximum independent set of those subsets and returns the conjunction of those independent sets. To divide the rectangles into sets horizontal lines $\ell_1, \ell_2, \ldots, \ell_m$ where $m \leq n$ are drawn such that the following three conditions hold:

- The distance between two adjacent lines is > 1. i.e. bigger than the height of a rectangle
- Each line intersects at least one rectangle
- Each rectangle is intersected by exactly one line

To find the correct positions for the lines, first sort the rectangles according to their vertical coordinates and draw the first line at the top of the rectangle with smallest y-coordinate and the following lines at the top of the rectangle with the smallest y-coordinate that is not yet intersected by a line. Figure 4a illustrates this. Now every rectangle has an intersecting line and the initial arrangement of rectangles is partitioned into sets $R_1, \ldots, R_m$ where for every $1 \leq i \leq m$ the set R_i contains those rectangles that are intersected by the line ℓ_i.

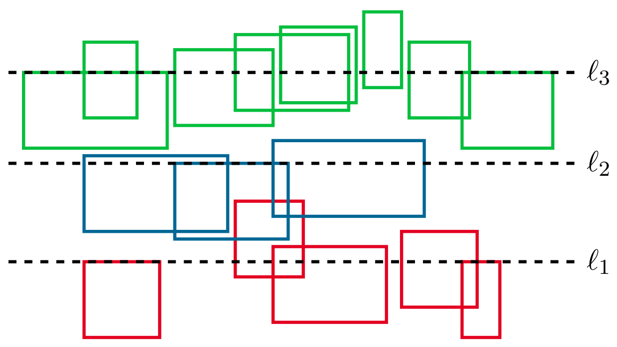
Due to the fact that the distance between two lines is bigger than the height of the rectangles a rectangle in the set R_i can only intersect with rectangles in the sets R_i and $R_i \pm 1$. Therefore the independent sets I_i of the even sets R_{2n} and the odd sets R_{2n+1} $(0 \geq n \geq {}^{m}\!/\!{}_{2})$ can be combined to two independent sets $I_{\text{odd}} = \bigcup_{n>0} I_{2n}$ and $I_{\text{even}} = \bigcup_{n \geq 0} I_{2n+1}$.

3.1.1 Greedy algorithm for MIS of intervals

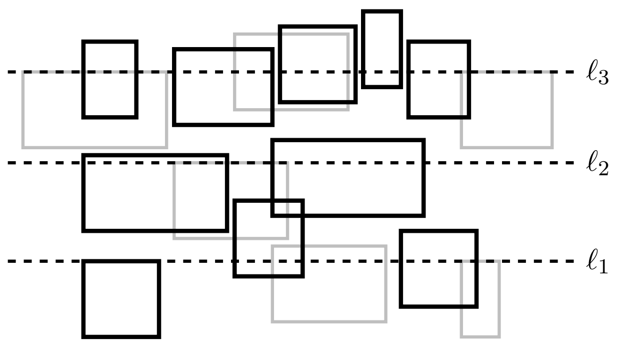
The MIS of every subset R_i can be found in polynomial time with a greedy algorithm. As can be seen in Figure 4c two rectangles that are intersected by a common horizontal line only intersect each other iff the intervals of their x-coordinates intersect each other. Therefore the problem of finding a MIS of the rectangles in one of the sets R_i can be reduced to finding a MIS of the horizontal projection of these rectangles. For this a greedy algorithm with runtime $\mathcal{O}(n \log n)$ exists. The idea of this algorithm is the following inductive argument: For every point p on the x-axis at most one interval in the MIS can be *active*, i.e. having a start point s and an end point e s.t. $s \leq p \leq e$. It follows that if two (or more) intervals intersect each other at one point at most one of these intervals can be added to an independent set. Choosing the interval with the lowest endpoint allows choosing more intervals after this interval (see Figure 4d).

3.1.2 Correctness and proof of approximation quality

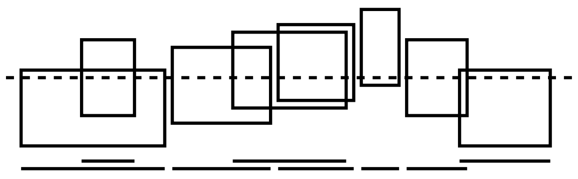
Theorem 5. *The set $I_{even} = \bigcup_{n>0} I_{2n}$ obtained by combining the maximum independent sets I_n of the sets R_n for even n is also a maximum independent set of $R_{even} = \bigcup_{n>0} R_{2n}$.*



(a) Unit height rectangles partitioned into three sets R_1, R_2, R_3 with intersecting lines ℓ_1, ℓ_2, ℓ_3.



(b) MIS I_1, I_2, I_3 (black) for the sets R_i. Note that rectangles from IS of two adjacent sets $(I_i, I_{i\pm1})$ may intersect each other.


(c) To find a MIS of R_3 only the x-coordinates have to be considered. Two rectangles intersect iff their projection to a horizontal line (below) intersects.

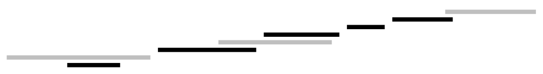

(d) Horizontal projection of rectangles in R_3 ordered (in vertical direction) by their endpoint. Greedy algorithm adds interval with nearest endpoint (bottom to top in this figure) that does not intersect rectangles already in the set.

Figure 4: Illustration of the 2-approximation algorithm for the unit height rectangle MIS problem.

Proof. The rectangles in this set are independent because a rectangle r from the set $I_x \subseteq R_x$ can't intersect any other rectangle in the set I_x because if this were the case I_x would not be an independent set. Furthermore the rectangle r can't intersect any rectangle $r' \in I_{\text{even}} \setminus I_x$ as r and r' both are intersected by a horizontal line with a vertical distance $2 + \varepsilon$ with $\varepsilon > 0$ and therefore the vertical distance between r and r' is at least ε. As every independent set I_x is the MIS of R_x the combination I_{even} is also the MIS of R_{even}. The same arguments apply to the set $I_{\text{odd}} \subseteq R_{\text{odd}}$.

□

The approximation quality guarantee follows from the fact that the MIS I_{OPT} of all the rectangles is at most $I_{\text{even}} + I_{\text{odd}}$. Therefore choosing the bigger of the sets I_{even} and I_{odd} leads to a 2-approximation in the case that $|I_{\text{even}}| = |I_{\text{odd}}|$. If one of the sets has more elements than the other the approximation is even better.

3.1.3 Runtime of 2-approximation algorithm

The runtime of the algorithm is dominated by the sorting of the rectangles which takes time $\mathcal{O}(n \log n)$. Partitioning the sorted rectangles with horizontal lines can be done in $\mathcal{O}(n)$ by iteratively taking the first (i.e. having the smallest y-coordinate) not-yet-intersected rectangle and adding a line at it's upper y-coordinate. Finding the MIS of every set R_i is also bounded by $\mathcal{O}(n \log n)$ resulting from the sorting the rectangles in each of the sets of the partition. Selecting the rectangles for the independent sets I_i, combining the even and odd independent sets and comparing their sizes can all be done in $\mathcal{O}(n)$.

3.2 Improving to a $(1 + 1/k)$-approximation algorithm

The algorithm described before separates the rectangles into independent subproblems for which efficient algorithms exist. The idea from Agarwal et al. was to improve this method by increasing the size of these subproblems in a way that still ensures that they can be solved in polynomial time for some fixed k. In the 2-approximation algorithm the MIS of R_n for either even or odd n were discarded completely. The idea of the improved algorithm is to partition the rectangles in the same way as before but to solve the MIS problem for sets of rectangles intersected by k consecutive lines. Those sets are referred to as *subgroups* and are defined in the following way:

$$R_i^k = \bigcup_{n=i}^{i+k-1} R_n.$$

For some value k there are $k+1$ *groups* $G_1, \ldots, G_{k+1}$ that correspond to the sets R_{odd} and R_{even}

$$G_j = R_1^{j-1} \cup \bigcup_{i \geq 0} R_{i(k+1)+j}^k = R \setminus \bigcup_{i \geq 0} R_{i(k+1)+j}.$$

The group G_j is therefore the set of all rectangles R except those intersected by every $(k+1)$-th line starting from the j-th line. Also all subgroups R_i^k in G_j are independent from each other, i.e. there are no two rectangles in two different subgroups that intersect each other. Computing the MIS $I_i \subseteq G_i$ for all $i \in \{1, \ldots, k+1\}$ and picking the biggest of those independent sets leads to a $(1 + {}^1\!/_k)$-approximate solution, as for every j the set $R \setminus G_j$ contains rectangles intersected by at most $\lceil {}^m\!/_{k+1} \rceil$ lines and therefore at most ${}^{|I_{\text{OPT}}|}\!/_{k+1}$ rectangles can be missed due to the pigeon hole principle.

To get the maximum independent set for the groups G_i a dynamic programming scheme is used that exceeds the scope of this report. In Agarwal et al. [1, Section 4.2] more details can be found.

References

[1] Pankaj K. Agarwal, Marc van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3–4):209–218, December 1998.

[2] J. L. Bentley and T. A. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Transactions on Computers*, C-28(9):643–647, September 1979.

[3] Nicolas Bourgeois, Bruno Escoffier, Vangelis Th Paschos, and Johan M. M. van Rooij. A Bottom-Up Method and Fast Algorithms for max independent set. In Haim Kaplan, editor, *Algorithm Theory - SWAT 2010*, number 6139 in Lecture Notes in Computer Science, pages 62–73. Springer Berlin Heidelberg, June 2010. DOI: 10.1007/978-3-642-13731-0_7.

[4] Andrew Bristow. The INDEPENDENT SET Decision Problem is NP-complete. *Theses and Dissertations*, August 2011.

[5] B. Chamaret, S. Josselin, P. Kuonen, M. Pizarroso, B. Salas-Manzanedo, S. Ubeda, and D. Wagner. Radio network optimization with maximum independent set search. In *Vehicular Technology Conference, 1997, IEEE 47th*, volume 2, pages 770–774 vol.2, May 1997.

[6] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A Measure & Conquer Approach for the Analysis of Exact Algorithms. *J. ACM*, 56(5):25:1–25:32, August 2009.

[7] Petr Hlineny and Jan Kratochvil. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Mathematics*, 229(1–3):101–124, February 2001.

[8] Tim Nieberg, Johann Hurink, and Walter Kern. A Robust PTAS for Maximum Weight Independent Sets in Unit Disk Graphs. In Juraj Hromkovič, Manfred Nagl, and Bernhard Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science*, number 3353 in Lecture Notes in Computer Science, pages 214–221. Springer Berlin Heidelberg, June 2004. DOI: 10.1007/978-3-540-30559-0_18.