Fixed-parameter algorithms for Cochromatic Number and Disjoint Rectangle Stabbing via Iterative Localization*

Pinar Heggernes† Dieter Kratsch ‡ Daniel Lokshtanov§ Venkatesh Raman¶
Saket Saurabh¶

Abstract

Given a permutation π of $\{1,\ldots,n\}$ and a positive integer k, can π be partitioned into at most k subsequences, each of which is either increasing or decreasing? We give an algorithm with running time $2^{O(k^2 \log k)}n^{O(1)}$ that solves this problem, thereby showing that it is fixed parameter tractable. This NP-complete problem is equivalent to deciding whether the cochromatic number of a given permutation graph on n vertices is at most k. Our algorithm solves in fact a more general problem: within the mentioned running time, it decides whether the cochromatic number of a given perfect graph on n vertices is at most k.

To obtain our result we use a combination of two well-known techniques within parameterized algorithms: iterative compression and greedy localization. Consequently we name this combination "iterative localization". We further demonstrate the power of this combination by giving an algorithm with running time $2^{O(k^2 \log k)}n \log n$ that decides whether a given set of n non-overlapping axis-parallel rectangles can be stabbed by at most k of a given set of horizontal and vertical lines.

1 Introduction

A *monotone subsequence* of a sequence of distinct integers is a collection of these integers that appear either in increasing or in decreasing order in the sequence. Given a permutation π on $[n] = \{1,\ldots,n\}$ and a positive integer k, a well known partitioning problem asks whether we can partition π into at most k monotone subsequences. This partition problem is NP-complete [31] and it can be solved in time $n^{O(k)}$ [3]. Using the famous Erdős-Szekeres theorem [15] which states that every sequence of $p \cdot q + 1$ real numbers has a monotone subsequence of length either $p+1$ or $q+1$, an algorithm with running time $n^{O(k)}$ implies a subexponential-time algorithm with running time $n^{O(\sqrt{n})}$ for partitioning π into the minimum number of monotone subsequences [3]. A natural question which has been left open, and most recently stated at a 2008 Dagstuhl Seminar [18], is whether the problem is fixed parameter tractable (FPT) when parameterized by the number of monotone subsequences. Equivalently, is there

*A preliminary version of this paper appeared in the proceedings of SWAT 2010. This work is supported by the Research Council of Norway and by ANR Blanc AGAPE.

†Department of Informatics, University of Bergen, Norway. pinar.heggernes@ii.uib.no

‡Université Paul Verlaine Metz, France. kratsch@univ-metz.fr

§Dept. of Comp. Sc. and Engin., University of California San Diego, USA. dlokshtanov@ucsd.edu

¶The Institute of Mathematical Sciences, Chennai, India. {vraman|saket}@imsc.res.in

an algorithm with running time $f(k)\ n^{O(1)}$ for partitioning a permutation into at most k monotone subsequences? We answer this question affirmatively by giving an algorithm with running time $2^{O(k^2 \log k)} n^{O(1)}$.

Every permutation π on $[n]$ corresponds to a *permutation graph* $G(\pi)$ on n vertices. This graph has a vertex for each integer $1, 2, \ldots, n$ and there is an edge between any two integers if their order in π is the opposite of their natural order, that is, for $i < j$, we have an edge between i and j if $\pi(i) > \pi(j)$. Hence, the above mentioned partitioning problem is equivalent to deciding whether the vertices of $G(\pi)$ can be partitioned into at most k independent sets or cliques. This brings us to the notion of cochromatic number of a graph. The *cochromatic number* of a graph $G = (V, E)$ is the minimum number of sets the vertex set V can be partitioned into, such that each set is either an independent set or a clique. Thus, the above mentioned partitioning problem is equivalent to finding the cochromatic number of permutation graphs. On arbitrary graphs, the Cochromatic Number problem is defined as follows.

Cochromatic Number
Input: A graph G on n vertices, and an integer $k \geq 1$.
Parameter: k.
Question: Is the cochromatic number of G at most k?

Cochromatic Number, being a natural extension of chromatic number and graph colorings, has been well studied [13, 14], and it is NP-complete even on permutation graphs [31]. Brandstädt [2] showed that we can recognize in polynomial time whether the vertex set of a given undirected graph can be partitioned into one or two independent sets and one or two cliques. However, it remains NP-complete to decide whether we can partition the given graph into at most κ independent sets and at most ℓ cliques if either $\kappa \geq 3$ or $\ell \geq 3$. It is easy to show that testing whether the cochromatic number of a given graph is at most 3 is NP-complete. Thus, we cannot hope to solve Cochromatic Number on general graphs even in time $n^{f(k)}$ for any arbitrary function f of k unless P=NP. We show that Cochromatic Number is fixed parameter tractable on *perfect graphs*; a graph class that subsumes permutation graphs, bipartite graphs, and chordal graphs, to name a few.

A graph is *perfect* if the chromatic number is equal to the clique number for each of its induced subgraphs. Perfect graphs were introduced by Berge in the early 60's, and they form one of the most well studied classes of graphs [4, 16, 22, 28]. Perfect graphs have many desirable algorithmic properties. They can be recognized in polynomial time [7], and one can find a maximum independent set, a minimum coloring, and a maximum clique, all in polynomial time [23]. Our algorithm solves Cochromatic Number in $2^{O(k^2 \log k)} n^{O(1)}$ time on perfect graphs and crucially uses several algorithmic properties of perfect graphs. To the best of our knowledge even an $n^{O(k)}$ time algorithm solving this problem on perfect graphs was not known before. The only known algorithmic results for the Cochromatic Number problem are by Fomin et al. [19] who gave a factor 1.71 approximation algorithm on comparability graphs and on cocomparability graphs, and a factor $\log n$ approximation algorithm on perfect graphs. Note that permutation graphs are both comparability and cocomparability, and these two graph classes are both perfect.

To show our result, we use a combination of two well-known techniques within parameterized algorithms, namely, iterative compression and greedy localization. Thus we term

this combination "iterative localization". This combination follows the well known iterative compression paradigm in parameterized complexity, but finds a small witness to branch and move towards an optimal solution "for the compressed instance" at the compression step.

Using this new combination, we are also able to resolve another open question [10, 11], namely whether Disjoint Rectangle Stabbing is fixed parameter tractable.

> Disjoint Rectangle Stabbing
> ***Input:*** A set R of n axis-parallel non-overlapping rectangles embedded in a plane, a set L of vertical and horizontal lines embedded in the plane, and an integer $k \geq 1$.
> ***Parameter:*** k.
> ***Question:*** Is there a set $L' \subseteq L$ with $|L'| \leq k$ such that every rectangle from R is stabbed by at least one line from L'?

Here we say that a rectangle is *stabbed* by a line if their intersection is nonempty. Furthermore, two rectangles are said to be *overlapping* if there exists a vertical line v and a horizontal line h such that both rectangles are stabbed by the lines v and h. In particular, non-intersecting rectangles are always non-overlapping.

The Rectangle Stabbing problem, the more general version of the Disjoint Rectangle Stabbing problem, where the rectangles can overlap is a generic geometric covering problem having wide applicability [24]. A number of polynomial-time approximation results for Rectangle Stabbing and its variants are known [8, 32, 26, 24]. In [11], the authors prove a W[1]-hardness result for a higher dimensional version of the Rectangle Stabbing problem and show several restrictions of this two dimensional version fixed-parameter tractable. Recently, Dom et al. [10] and Giannopoulos et al. [21] independently considered the general two dimensional version and showed it to be complete for the parameterized complexity class W[1]. They also showed a restricted version of Disjoint Rectangle Stabbing, b-Disjoint Square Stabbing fixed parameter tractable for a fixed b. All these papers leave the parameterized complexity of the general Disjoint Rectangle Stabbing problem open.

Our paper is organized as follows. In Section 2 we give the remaining necessary definitions and set up our notations. Section 3 gives an overview of the method we use to solve the two problems we address. The fixed parameter tractable algorithm for Cochromatic Number on perfect graphs is given in Section 4 and for Disjoint Rectangle Stabbing in Section 5. We conclude with some remarks in Section 6.

2 Definitions and Notation

All graphs in this paper are simple, undirected, and unweighted. For a graph $G = (V, E)$, we denote the size of the vertex set V by n and the edge set E by m. For a subset S of V, the *subgraph of G induced by S* is denoted by $G[S]$. The *complement* of $G = (V, E)$ is denoted by $\overline{G}$, it has the same vertex set V, and edge set $\{uv \mid u, v \in V \text{ and } u \neq v \text{ and } uv \notin E\}$.

A *(proper) coloring* of a graph is an assignment of colors to its vertices so that adjacent vertices receive different colors. A coloring is *minimum* if it uses the minimum number of colors. The *chromatic number* and the *clique number* of G are, respectively, the number of

colors in a minimum coloring of G, and the size of a largest clique in G. A *clique cover* in a graph is a collection of cliques such that every vertex is in one of the cliques of the collection. A graph is *perfect* if the chromatic number is equal to the clique number for each of its induced subgraphs. The *cochromatic number* of a graph G is the minimum number of sets V can be partitioned into, such that each set is either an independent set or a clique.

A parameterized problem L takes two values as input – an input x and an integer parameter k, and is said to be fixed parameter tractable (FPT) if there is an algorithm that decides whether the input (x, k) is in L in $f(k)n^{O(1)}$ time where f is some function of k. We refer to [12, 29, 17] for more information on fixed-parameter algorithms and parameterized complexity.

3 Methodology

Iterative Localization can be viewed as a combination of two well known methods in obtaining fixed parameter tractable algorithms, that is, *greedy localization* and *iterative compression*. The method of greedy localization is primarily used for maximization problems. The idea is to start off by greedily computing a solution to the problem at hand and showing that the optimal solution must in some sense lie "close" to the current solution. For a concrete example consider the problem of finding k vertex disjoint P_3's, paths on three vertices, in a given graph G. We greedily compute a maximal collection of pairwise disjoint P_3's. If the number of P_3's in our collection is at least k, we are done. Else, observe that any collection of k pairwise disjoint P_3's must intersect with vertices of P_3's in our greedy solution. We refer to [9, 25] for applications of greedy localization.

Iterative Compression is a technique primarily used for minimization problems. Algorithms based on iterative compression apply polynomially many "compression steps". In a compression step, we are given an instance I of the problem, a solution S' to the problem, and the objective is to check whether there exists a solution S for I such that $|S| < |S'|$ in $f(|S'|)|I|^{O(1)}$ time. The idea is to process the instance incrementally, maintaining a solution S to the intermediate instances. In each incremental step both the instance and the maintained solution increase in size. Then the compression algorithm is run to decrease the size of S again. Iterative Compression has proved very useful in the design of parameterized algorithms and is instrumental in the currently fastest parameterized algorithms for Odd Cycle Transversal [30], Feedback Vertex Set [6] and Directed Feedback Vertex Set [5]. We refer to [29] for a more thorough introduction to Iterative Compression.

We combine the two methods, applying iterative compression to incrementally build the solution. In each compression step we search the solution space around the given solution similar to how it is done in greedy localization.

4 Cochromatic Number on Perfect Graphs

In this section we give an algorithm for Cochromatic Number on perfect graphs. We start by guessing the number α of cliques and the number β of independent sets such that $\alpha + \beta = k$, and for each of the $k + 1$ guesses (of (α, β)) we will decide whether G can be partitioned into at most α cliques and at most β independent sets. We order V into

$v_1, v_2, \ldots, v_n$ and define $V_i = \{v_1, \ldots, v_i\}$ for every i. Notice that if V can be partitioned into at most α cliques and β independent sets, then for every i, so can $G[V_i]$. Also, given such a partitioning for $G[V_i]$ we can easily make a partitioning of $G[V_{i+1}]$ into $\alpha + 1$ cliques and β independent sets by letting v_{i+1} be a clique by itself. At this point we want to use the current partitioning to decide whether there is a partitioning of $G[V_{i+1}]$ into α cliques and β independent sets. This naturally leads to the definition of the compression version of the problem.

> Compression Cochromatic Number (CCN)
> ***Input:*** A perfect graph $G = (V, E)$ on n vertices, a partition $\mathcal{P} = (C_1, \ldots, C_{\alpha+1}, I_1, \ldots, I_\beta)$ of V, where C_i, $1 \leq i \leq \alpha + 1$, are cliques, and I_j, $1 \leq j \leq \beta$, are independent sets.
> ***Task:*** Find a partitioning of G into α cliques and β independent sets, or conclude that no such partitioning exists.

We will give a $2^{O(\alpha\beta\log(\alpha\beta))}n^{O(1)}$ time algorithm for CCN, which together with the discussion above yields a $2^{O(\alpha\beta\log(\alpha\beta))}n^{O(1)}$ time algorithm for Cochromatic Number. To do that we use the following classical results.

Lemma 1 ([23])**.** *Let G be a perfect graph. Then there exist algorithms that can compute in polynomial time: (a) a minimum coloring of G, (b) a maximum independent set of G, and (c) a maximum clique of G.*

Lemma 2 ([27])**.** *G is a perfect graph if and only if $\overline{G}$ is a perfect graph.*

Using Lemmata 1 and 2, we can prove the following preliminary lemma which is integral to our algorithm.

Lemma 3. *Given a perfect graph $G = (V, E)$ and an integer ℓ, there is a polynomial time algorithm that outputs*

(a) either a partition of V into at most ℓ independent sets or a clique of size $\ell + 1$, and

(b) either a partition of V into at most ℓ cliques or an independent set of size $\ell + 1$.

Proof. We first give a proof for (a). We start by finding a minimum coloring of G in polynomial time using Lemma 1. If the number of colors required is at most ℓ, then we have our required partitioning of V into at most ℓ independent sets. Otherwise since G is a perfect graph, the chromatic number of G is the same as the clique number of G. Hence a maximum clique is of size at least $\ell + 1$. Now we find a maximum clique using Lemma 1. Let C be such a clique. Now choose an arbitrary subset $C' \subseteq C$ of size $\ell + 1$ and return C' as the desired clique of size $\ell + 1$. To prove part (b) we just need to observe that a clique in G is an independent set in $\overline{G}$. Now the proof follows by the proof of (a) and using the fact that $\overline{G}$ is also a perfect graph (by Lemma 2). □

We will now explain how the given partitioning $\mathcal{P}$ is useful in solving the compression step. The main idea is that while the partitioning $\mathcal{P}'$ we search for may differ significantly from $\mathcal{P}$, only a few vertices that were covered by cliques in $\mathcal{P}$ can be covered by independent

sets in $\mathcal{P}'$ and vice versa. To formalize this idea we introduce the notion of a bitstring $B_{\mathcal{P}}$ of the partition $\mathcal{P}$.

Given $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$, and a partition $\mathcal{P}$ of V into cliques and independent sets, let $B_{\mathcal{P}}$ be the binary vector of length n in which position i is 0 if v_i belongs to a clique in $\mathcal{P}$, and 1 if v_i belongs to an independent set in $\mathcal{P}$. Given a n-vertex graph G and a bitstring B of length n, define X_B to be the set of vertices of G whose corresponding entry in B is 0, and $Y_B = V \setminus X_B$. For two integers α and β we say that B is *valid* in G with respect to α and β if there exists a partition $\mathcal{P}$ of V into at most α cliques and at most β independent sets such that $B = B_{\mathcal{P}}$. Given two bitstrings B_1 and B_2 of equal length, the *hamming distance* between B_1 and B_2 is the number of positions on which the corresponding bits differ, and it is denoted by $\mathcal{H}(B_1, B_2)$.

First, in Lemma 4 we will show that for a perfect graph G a valid bitstring B is sufficient to reconstruct a partition of G into α cliques and β independent sets. Then, in Lemma 5 we will show that two valid bitstrings must be "similar".

Lemma 4. *There is a polynomial time algorithm that, given a perfect graph $G = (V, E)$ on n vertices, a bitstring B of length n, and positive integers α and β, tests whether B is valid in G with respect to α and β. If B is valid the algorithm outputs a partition $\mathcal{P}$ of V into α cliques and β independent sets. If not, the algorithm either outputs an independent set of size $\alpha + 1$ in $G[X_B]$ or a clique of size $\beta + 1$ in $G[Y_B]$.*

Proof. As G is perfect, $G[X_B]$ and $G[Y_B]$ the induced subgraphs on X_B and Y_B respectively are perfect. The algorithm first uses Lemma 3 (b) to either find a partitioning of $G[X_B]$ into α cliques or an independent set of size $\alpha + 1$ in $G[X_B]$. Then it uses Lemma 3 (a) to either find a partitioning of $G[Y_B]$ into β independent sets or a clique set of size $\beta + 1$ in $G[Y_B]$. □

Lemma 5. *Let $G = (V, E)$ be a graph, $\mathcal{P} = (C_1, C_2, \ldots, C_\alpha, I_1, \ldots, I_\beta)$ be a partition of G into α cliques and β independent sets and $\mathcal{Q} = (C'_1, C'_2, \ldots, C'_{\alpha'}, I'_1, \ldots, I'_{\beta'})$ be a partition of G into α' cliques and β' independent sets. Let $B_{\mathcal{P}}$ and $B_{\mathcal{Q}}$ be the bitstrings associated $\mathcal{P}$ and $\mathcal{Q}$ respectively. Then $\mathcal{H}(B_{\mathcal{P}}, B_{\mathcal{Q}}) \leq \alpha\beta' + \alpha'\beta$.*

Proof. Observe that an independent set and a clique can intersect in at most one vertex. Hence

$$\left| \left(\bigcup_{i=1}^{\alpha} C_i \right) \bigcap \left(\bigcup_{j=1}^{\beta'} I'_j \right) \right| \leq \alpha\beta' \qquad \text{and} \qquad \left| \left(\bigcup_{i=1}^{\beta} I_i \right) \bigcap \left(\bigcup_{j=1}^{\alpha'} C'_j \right) \right| \leq \beta\alpha'.$$

This concludes the proof of the lemma. □

We are now ready to present our algorithm for the compression step. Recall that we are given a perfect graph G as input, together with integers α and β and a partition $\mathcal{P}$ of G into $\alpha + 1$ cliques and β independent sets. The task is to find a partition $\mathcal{P}'$ of G into α cliques and β independent sets, if such a partition exists. Lemma 5 yields that it is sufficient to look for solutions with bitstrings 'close to' $B_{\mathcal{P}}$. Lemma 3 is used to pinpoint the "wrong" bits of $B_{\mathcal{P}}$. Formally, the algorithm Algo-CCN takes as input a perfect graph G, a bitstring B and integers α, β and μ. It outputs a partition $\mathcal{P}'$ of G into α cliques and β independent sets such that $\mathcal{H}(B_{\mathcal{P}'}, B) \leq \mu$ if such a partition exists, and answers "NO" otherwise. To

ALGO-CCN$(G, B, \alpha, \beta, \mu))$
(Here G is a perfect graph, B is a bit vector, α, β and μ are integers. ALGO-CCN outputs a partition $\mathcal{P}'$ of G into α cliques and β independent sets such that $\mathcal{H}(B_{\mathcal{P}'}, B) \leq \mu$ if such a partition exists, or answers "NO" otherwise.)

1. If $\mu < 0$ return "NO".
2. Use Lemma 4 to either find a partition of $\mathcal{P}'$ of G into α cliques and β independent sets with $B_{\mathcal{P}'} = B$ or find an independent set $I \subseteq X_B$ of size $\alpha + 1$ or find a clique $C \subseteq Y_B$ of size $\beta + 1$. If a partition was found answer "YES".
3. If an independent set I was found in step 2: For each vertex $v \in I$, let the bitvector $B(v)$ be obtained from B by flipping v's bit 0 to 1. For each $v \in I$, recursively solve the subproblem ALGO-CCN$(G, B(v), \alpha, \beta, \mu - 1)$. Return "YES" if any of the recursive calls returns "YES", otherwise return "NO".
4. If a clique C was found in step 2: For each vertex $v \in C$, let the bitvector $B(v)$ be obtained from B by flipping v's bit 1 to 0. For each $v \in C$, recursively solve the subproblem ALGO-CCN$(G, B(v), \alpha, \beta, \mu - 1)$. Return "YES" if any of the recursive calls returns "YES", otherwise return "NO".

Figure 1: Description of Algorithm ALGO-CCN

solve the problem CCN we call ALGO-CCN$(G, B_{\mathcal{P}}, \alpha, \beta, 2\alpha\beta + \beta)$. A formal description of the algorithm Algo-CCN is given in Figure 1.

Lemma 6. *The call to* ALGO-CCN$(G, B_{\mathcal{P}}, \alpha, \beta, 2\alpha\beta + \beta)$ *correctly solves the* CCN *instance* G,$\mathcal{P}$,α,β *in time* $(\alpha + \beta)^{2\alpha\beta+\beta} n^{O(1)}$.

Proof. We first argue about the correctness. By Lemma 5 it is sufficient to search for partitions $\mathcal{P}'$ such that $\mathcal{H}(B_{\mathcal{P}'}, B_{\mathcal{P}}) \leq 2\alpha\beta + \beta$. If the algorithm answers "YES" it also outputs a partition $\mathcal{P}'$ of G into α cliques and β independent sets such that $B_{\mathcal{P}'}$ was obtained from $B_{\mathcal{P}}$ by flipping at most $2\alpha\beta + \beta$ bits. Thus it remains to argue that if such a partition $\mathcal{P}'$ exists, the algorithm will find it. Let $B' = B_{\mathcal{P}'}$.

In any recursive call, if there exists an independent set I of size $\alpha + 1$ in $G[X_B]$ then there is a vertex $v \in I$ whose corresponding bit is 1 in B'. The algorithm tries flipping the bit of each v in I, decreasing the hamming distance between B and B' by one in at least one recursive call. Similarly, if there exists a clique C of size $\beta + 1$ in $G[Y_B]$ then there is a vertex $v \in C$ whose corresponding bit is 0 in B'. Again, the algorithm tries flipping the bit of each v in C, decreasing the hamming distance between B and B' by one in at least one recursive call, and the correctness of the algorithm follows.

To argue the time bound, we consider a slight modification of the algorithm that if either $\alpha = 0$ or $\beta = 0$, applies Lemma 4 to solve the CCN instance in polynomial time. Hence without loss of generality $\alpha + 1 \leq \alpha + \beta$ and $\beta + 1 \leq \alpha + \beta$. Then every node in the branch tree has at most $\alpha + \beta$ children and the depth of the tree is at most $2\alpha\beta + \beta$ and hence the number of nodes in the branch tree is $O((\alpha + \beta)^{2\alpha\beta+\beta})$. Since the amount of work in each

node is polynomial, the time bound follows. □

Now we are ready to prove the main theorem of this section.

Theorem 1. *The* Cochromatic Number *problem can be solved in* $2^{O(k^2 \log k)} n^{O(1)}$ *time on perfect graphs.*

Proof. We apply iterative compression as described in the beginning of this section. In particular, the algorithm guesses the value of α and β and decides whether G can be partitioned into α cliques and β independent sets using $n-k-1$ compression steps. The i'th compression step is resolved by calling Algo-CCN$(G[V_i], B_{\mathcal{P}}, \alpha, \beta, 2\alpha\beta+\beta)$ where $\mathcal{P}$ is the partition into $\alpha+1$ cliques and β independent sets. The correctness and time bound follow from Lemma 6. □

5 Disjoint Rectangle Stabbing

In this section we give a fixed parameter tractable algorithm for Disjoint Rectangle Stabbing. Recall that a rectangle is stabbed by a line if their intersection is nonempty. In $O(n \log n)$ time we can sort the coordinates of the rectangles in non-decreasing order, and make two lists containing all the rectangles, one with the rectangles sorted in non-decreasing order by their x-coordinates and the other where the rectangles are sorted in non-decreasing order by their y-coordinates of their top right corner. We also sort the set L_H of horizontal lines in L by their y-coordinates and the set L_V of vertical lines in L by their x-coordinates. *Whenever we speak of a subset of R (the set of all rectangles) or L we will assume that the corresponding sorted lists are given.*

For each of the $k+1$ possible choices of non-negative integers α, β such that $\alpha+\beta=k$ we will run an algorithm that decides whether the rectangles can be stabbed by at most α horizontal and β vertical lines. In order to get an $O(n \log n)$ time bound for fixed k, we apply a *recursive* compression scheme rather than an iterative compression scheme. In particular, our algorithm partitions the n rectangles into two groups R_1 and R_2 with at most $\lceil n/2 \rceil$ rectangles in each group and runs recursively on the two groups. Now, if R can be stabbed by α horizontal and β vertical lines then so can R_1 and R_2, so if either of the recursive calls returns "NO" we return "NO" as well. Otherwise we combine the solutions to R_1 and R_2 to a solution with at most 2α horizontal and at most 2β vertical lines that stab R. We want to use this solution in order to decide whether R can be stabbed by at most α horizontal and at most β vertical lines. This leads to the definition of the compression version of Disjoint Rectangle Stabbing.

> Compression Disjoint Rectangle Stabbing (CDRS)
> ***Input:*** A set R of n axis-parallel non-overlapping rectangles embedded in the plane, a set $L = L_H \cup L_V$ of horizontal and vertical lines embedded in the plane, integers α and β and a set $Z \subseteq L$ with at most 2α horizontal and at most 2β vertical lines such that every rectangle from R is stabbed by at least one line from Z.
> ***Task:*** Find a set $L' \subseteq L$ with at most α horizontal and at most β vertical lines such that every rectangle from R is stabbed by at least one line from L'. If no such set exists, return "NO".

The algorithm for CDRS has exactly the same structure as Algo-CCN, but with the individual pieces tailored to fit the Disjoint Rectangle Stabbing problem. We start by proving a lemma analogous to Lemma 3.

Lemma 7. *Given a set R of axis-parallel rectangles in the plane, two sets L_V, L_H of vertical and horizontal lines respectively, and an integer k, there is an $O(n)$ time algorithm that outputs*

(a) either a set $L'_H \subseteq L_H$ of lines with $|L_H| \leq k$ that stabs all rectangles in R or a collection H of $k+1$ rectangles such that each horizontal line in L stabs at most one rectangle in H, and

(b) either a set $L'_V \subseteq L_V$ of lines with $|L_V| \leq k$ that stabs all rectangles in R or a collection V of $k+1$ rectangles such that each vertical line in L stabs at most one rectangle in V.

Proof. We only show (b) as the proof of (a) is identical. Initially $V = \emptyset$ and $L'_V = \emptyset$. We process R sorted in non-decreasing order of the x-coordinate of the top-right corner. At any step, if the considered rectangle r is not stabbed by any line in L'_V, add r to V and add the rightmost vertical line in L stabbing r to L'_V (to check whether r is not stabbed by any line in L'_V, we just keep track of the right most line in L'_V and check whether it stabs r; to find the right most vertical line in L that stabs r, we compute and keep this for every r in a preprocessing step). If at any step $|V| \geq k+1$, output V otherwise all rectangles have been considered, $|L'_V| \leq k$ and every rectangle is stabbed by some line in L_V. Clearly no vertical line in L can stab any two rectangles in V. The described algorithm can easily be implemented to run in $O(n)$ time. □

Now we define the notion of a bitstring of a solution and prove lemmata analogous to Lemmata 4 and 5 for CDRS. Let $L' \subseteq L$ be a set of lines that stab all the rectangles of R. We define the bitstring $B_{L'}$ of L' as follows. Let R be ordered into $r_1, r_2, \ldots, r_n$. The i'th bit of $B_{L'}$ is set to 0 if r_i is stabbed by a horizontal line in L' and 1 otherwise. Observe that if r_i is stabbed both by a horizontal and by a vertical line of L', the i'th bit of $B_{L'}$ is 0. Given a collection R of rectangles, a set L of horizontal and vertical lines, integers α and β and a bitstring B of length n, we say that B is *valid in R with respect to α and β* if there exists a set $L' \subseteq L$ of at most α horizontal lines and at most β vertical lines stabbing all rectangles in R such that $B = B_{L'}$. Given R and B we define X_B to be the set of rectangles in R whose corresponding entry in B is 0, and $Y_B = R \setminus X_B$.

Lemma 8. *There is an $O(n)$ time algorithm that given a collection R of n axis-parallel rectangles, a collection L of horizontal and vertical lines, a bitstring B of length n, and positive integers α and β, tests whether B is valid in R with respect to α and β. If B is valid the algorithm outputs a set $L' \subseteq L$ of at most α horizontal and at most β vertical lines such that every rectangle in R is stabbed by a line in L' and $B_{L'} = B$. If B is not valid, the algorithm either outputs a set $H \subseteq R$ of size $\alpha+1$ such that every horizontal line in L stabs at most 1 rectangle in H or a set $V \subseteq R$ of size $\beta+1$ such that every vertical line in L stabs at most 1 rectangle in V.*

Proof. Let X_B be the set of rectangles in R whose bit in B is 0, and let $Y_B = R \setminus X_B$. Apply the first part of Lemma 7 to X_B and the second part of Lemma 7 to Y_B. □

Now we are ready to show an analogue of Lemma 5 to the CDRS problem. To that end, for a line $l \in L$ let S_l be the set of rectangles in R stabbed by L. The proof of Lemma 9 relies on the fact that the rectangles in R are non-overlapping.

Lemma 9. *Let R be a collection of n non-overlapping axis-parallel rectangles, L be a collection of horizontal and vertical lines, $P \subseteq L$ and $Q \subseteq L$ be sets of lines so that every $r \in R$ is stabbed by a line in P and a line in Q. Suppose $|P \cap L_H| \leq \alpha$, $|P \cap L_V| \leq \beta$, $|Q \cap L_H| \leq \alpha'$ and $|Q \cap L_V| \leq \beta'$. Let B_P be the bitstring of P and B_Q be the bitstring of Q. Then $\mathcal{H}(B_P, B_Q) \leq \alpha\beta' + \alpha'\beta$.*

Proof. Notice that since the rectangles are non-overlapping, if $l_H \in L_H$ and $l_V \in L_V$ then $|S_{l_H} \cap S_{l_V})| \leq 1$. Hence

$$\left| \left(\bigcup_{p \in P \cap L_H} S_p \right) \bigcap \left(\bigcup_{q \in Q \cap L_V} S_q \right) \right| \leq \alpha\beta' \text{ and } \left| \left(\bigcup_{p \in P \cap L_V} S_p \right) \bigcap \left(\bigcup_{q \in Q \cap L_H} S_q \right) \right| \leq \alpha'\beta.$$

This concludes the proof of the lemma. □

Now we have given all the ingredients required to solve the compression step of the disjoint rectangle stabbing problem. The proof of the following Lemma is identical to the proof of Lemma 6.

Lemma 10. *The* Compression Disjoint Rectangle Stabbing *problem can be solved in $O((\alpha + \beta)^{4\alpha\beta + O(1)} n)$ time.*

Theorem 2. *The* Disjoint Rectangle Stabbing *problem can be solved in $2^{O(k^2 \log k)} n \log n$ time.*

Proof. The algorithm follows the recursive scheme described in the beginning of this section and uses the algorithm of Lemma 10 to solve the compression step. Correctness follows directly from Lemma 10. Let $T(n, k)$ be the time required to solve an instance with n rectangles and $\alpha + \beta = k$. Then

$$T(n, k) \leq 2T(n/2, k) + 2^{O(k^2 \log k)} n$$

which solves to $T(n, k) \leq 2^{O(k^2 \log k)} n \log n$ by the Master Theorem. The $O(k)$ overhead of trying all possible values of α and β with $\alpha + \beta = k$ is subsumed in the asymptotic notation. □

In [10], it is observed that the version of the Rectangle Stabbing problem, where we are given only the set of rectangles and we need to find at most k horizontal and vertical lines to stab them, polynomially reduces to the version where we are also given a set of horizontal and vertical lines. Hence this version of the Disjoint Rectangle Stabbing problem (where we are not given a set of lines) is also fixed parameter tractable.

Remark: It is worth noting that while the polynomial factor of our algorithm for Cochromatic Number of perfect graphs is rather large, it is only $n \log n$ for Disjoint Rectangle Stabbing. In fact our algorithm for Cochromatic Number can be made to run in time $2^{O(k^2 \log k)} n \log n$ on permutation graphs if the permutation model is given. In a permutation

graph, independent sets are increasing sequences in the permutation model while cliques are decreasing sequences in the model [20]. For a permutation of length n and an integer k, a simple greedy algorithm will in time $O(nk)$ either partition the permutation into k increasing sequences or find a decreasing sequence of length $k+1$. Thus, for permutation graphs one can make the algorithm of Lemma 4 run in time $O(nk)$ yielding a $2^{O(k^2 \log k)}n$ time bound for the compression step. Combining this with the same recursive scheme as the one in Theorem 2 one obtains the desired $2^{O(k^2 \log k)}n \log n$ running time bound.

6 Conclusion

Using a new combination of iterative compression and greedy localization we have established that two interesting problems in different domains, Cochromatic Number on perfect graphs and Disjoint Rectangle Stabbing, are fixed parameter tractable. The methodology is applicable whenever the $(k+1)$-sized solution at the beginning of a compression step of the iterative compression technique can be shown to be not 'too far' (in some specific sense) from a k-sized solution if exists, and if we can find in polynomial (or even fixed parameter tractable) time, a small witness to branch on in the compression step. It would be interesting to explore further applications of this combination.

References

[1] C. Berge, *Färbung von Graphen, deren sämtliche bzw. deren ungeraden Kreise starr sind (Zusammenfassung)*, Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur., Reihe 10 (1961) 114.

[2] A. Brandstädt, *Partitions of graphs into one or two independent sets and cliques*, Discrete Mathematics, 152 (1996) 47–54.

[3] A. Brandstädt and D. Kratsch, *On the partition of permutations into increasing or decreasing subsequences*, Elektron. Inform. Kybernet., 22 (1986), 263–273.

[4] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes: A Survey*, SIAM, 1999.

[5] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon, *A fixed-parameter algorithm for the directed feedback vertex set problem*, J. ACM, 55 (2008).

[6] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger, *Improved algorithms for feedback vertex set problems*, J. Comput. Syst. Sci., 74 (2008) 1188–1198.

[7] M. Chudnovsky, G. Cornuejols, X. Liu, P. Seymour, and K. Vuskovic, *Recognizing berge graphs*, Combinatorica, 25 (2005) 143–186.

[8] T. I. D. R. Gaur and R. Krishnamurti, *Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem*, Journal of Algorithms, 43 (2002) 138–152.

[9] F. K. H. A. Dehne, M. R. Fellows, F. A. Rosamond, and P. Shaw, *Greedy localization, iterative compression, modeled crown reductions: New FPT techniques, an improved algorithm for set splitting, and a novel 2k kernelization for vertex cover*, in IWPEC, vol. 3162 of Springer LNCS (2004) 271–280.

[10] M. Dom, M. R. Fellows, and F. A. Rosamond, *Parameterized complexity of stabbing rectangles and squares in the plane*, in WALCOM, vol. 5431 of Springer LNCS (2009) 298–309.

[11] M. Dom and S. Sikdar, *The parameterized complexity of the rectangle stabbing problem and its variants*, in FAW, vol. 5059 of Springer LNCS (2008) 288–299.

[12] R. D. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999.

[13] P. Erdős and J. Gimbel, *Some problems and results in cochromatic theory*, in Quo Vadis, Graph Theory?, North-Holland, Amsterdam, (1993) 261–264.

[14] P. Erdős, J. Gimbel, and D. Kratsch, *Extremal results in cochromatic and dichromatic theory*, Journal of Graph Theory, 15 (1991) 579–585.

[15] P. Erdős and G. Szekeres, *A combinatorial problem in geometry*, Compositio Mathematica, 2 (1935) 463–470.

[16] P. C. Fishburn, *Interval Orders and Interval Graphs: A Study of Partially Ordered Sets*, Wiley, 1985.

[17] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer-Verlag, 2006.

[18] F. V. Fomin, K. Iwama, D. Kratsch, P. Kaski, M. Koivisto, L. Kowalik, Y. Okamoto, J. van Rooij, and R. Williams, *08431 open problems – moderately exponential time algorithms*, in Moderately Exponential Time Algorithms, F. V. Fomin, K. Iwama, and D. Kratsch, eds., no. 08431 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.

[19] F. V. Fomin, D. Kratsch, and J.-C. Novelli, *Approximating minimum cocolorings*, Inf. Process. Lett., 84 (2002) 285–290.

[20] A. Frank, *On chain and antichain families of a partially ordered set*, J. Comb. Theory, Ser. B, 29 (1980) 176–184.

[21] P. Giannopoulos, C. Knauer, G. Rote, and D. Werner, *Fixed-parameter tractability and lower bounds for stabbing problems*, in Proceedings of the 25th European Workshop on Computational Geometry (EuroCG), (2009) 281–284.

[22] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, vol. 57, Elsevier, 2004. Second edition.

[23] M. Grötschel, L. Lovász, and A. Schrijver, *Polynomial algorithms for perfect graph*, Annals of Discrete Mathematics, 21 (1984) 325–356.

[24] R. Hassin and N. Megiddo, *Approximation algorithms for hitting objects with straight lines*, Discrete Applied Mathematics, 30 (1991) 29–42.

[25] W. Jia, C. Zhang, and J. Chen, *An efficient parameterized algorithm for -set packing*, J. Algorithms, 50 (2004) 106–117.

[26] S. Kovaleva and F. C. R. Spieksma, *Approximation algorithms for rectangles tabbing and interval stabbing problems*, SIAM J. Discrete Mathematics, 20 (2006) 748–768.

[27] L. Lovász, *A characterization of perfect graphs*, J. Comb. Theory, Ser. B, 13 (1972) 95–98.

[28] N. Mahadev and U. Peled, *Threshold graphs and related topics*, vol. 56, North Holland, 1995.

[29] R. Niedermeier, *Invitation to Fixed Parameter Algorithms*, Oxford Lecture Series in Mathematics and Its Applications, Oxford University Press, USA, 2006.

[30] B. A. Reed, K. Smith, and A. Vetta, *Finding odd cycle transversals*, Oper. Res. Lett., 32 (2004) 299–301.

[31] K. Wagner, *Monotonic coverings of finite sets*, Elektron. Inform. Kybernet., 20 (1984) 633–639.

[32] G. Xu and J. Xu, *Constant approximation algorithms for rectangle stabbing and related problems*, Theory of Computing Systems, 40 (2007) 187–204.