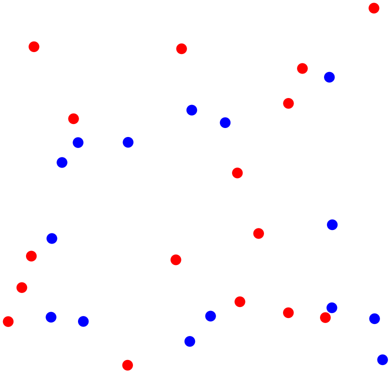
On the Parameterized Complexity of Red-Blue Points Separation

Édouard Bonnet, Panos Giannopoulos, and Michael Lampis

Middlesex University, London

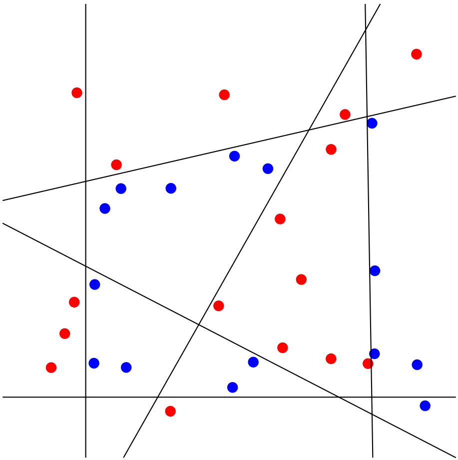
September 6, 2017, IPEC, Vienna

Red-Blue Separation



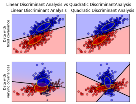
Given a set $\mathcal{R}$ of red points and $\mathcal{B}$ of blue points...

Red-Blue Separation



...find at most k lines making each cell monochromatic.

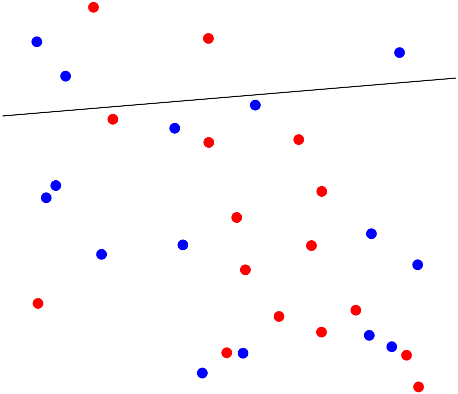
A few words on the problem



- motivated by machine learning applications
- natural geometric separation problem
- NP-hard [Meggido '88]
- APX-hard for the variant with axis-parallel lines...
- ...with an LP-based 2-approximation [Calinescu et al. '05]
- we will forbid the selected lines to contain any input point

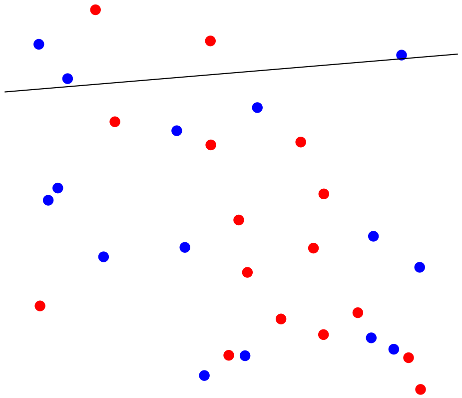
Discretization of the problem

There is an easy $(4n^2)^k = n^{O(k)}$-time algorithm



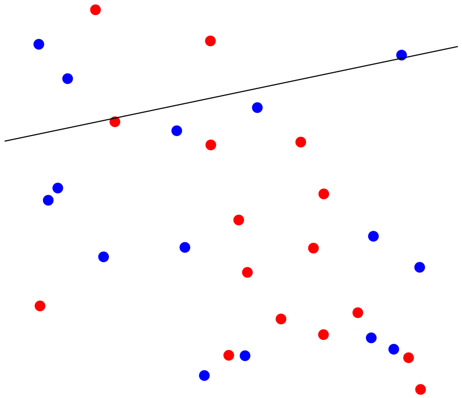
Discretization of the problem

There is an easy $(4n^2)^k = n^{O(k)}$-time algorithm



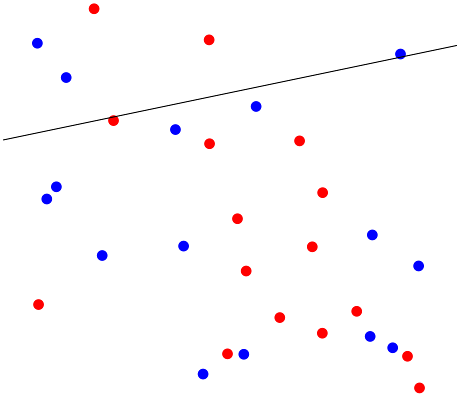
Discretization of the problem

There is an easy $(4n^2)^k = n^{O(k)}$-time algorithm



Discretization of the problem

There is an easy $(4n^2)^k = n^{O(k)}$-time algorithm



$O(n)$ for $k = 1$ and $O(n \log n)$ for $k = 2$ [Hurtado et al. '04].

Parameterized complexity?

A line arrangement created by k lines has $O(k^2)$ cells.
YES-instances are well-strucured:
decomposable into $f(k) = O(k^2)$ convex monochromatic regions.

Parameterized complexity?

A line arrangement created by k lines has $O(k^2)$ cells.
YES-instances are well-strucured:
decomposable into $f(k) = O(k^2)$ convex monochromatic regions.

Maybe FPT algorithm based on a kernel...

Parameterized complexity?

A line arrangement created by k lines has $O(k^2)$ cells.
YES-instances are well-strucured:
decomposable into $f(k) = O(k^2)$ convex monochromatic regions.

Maybe FPT algorithm based on a kernel...

Our main result

Red-Blue Separation cannot be solved in time $f(k)n^{o(k/\log k)}$ unless the ETH fails.

It almost matches the brute-force $n^{O(k)}$:

Red-Blue Separation is *not* part of those geometric problems solvable in $n^{O(\sqrt{k})}$.

Intermediate problems worth knowing

...to design parameterized lower bounds of geometric problems

- Grid Tiling [Marx '05] $\rightarrow$ no $f(k)n^{o(\sqrt{k})}$ for several geometric packing and covering problems
- Many classical optimisation problems on multiple-interval graphs [Jiang '10, Jiang and Zhang '12]: typically W[1]-hardness on (unit) 2-track interval and (unit) 2-interval graphs

Intermediate problems worth knowing

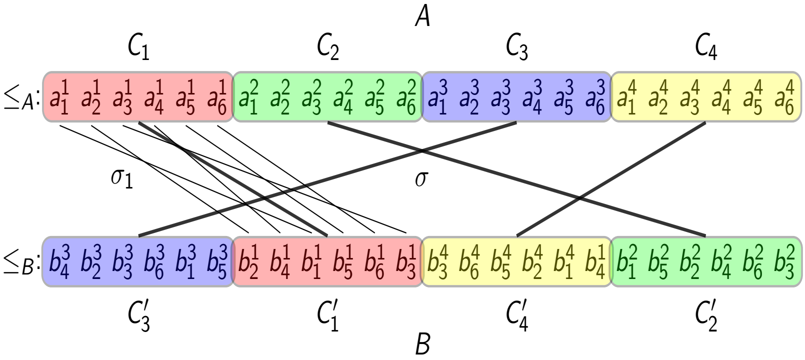
...to design parameterized lower bounds of geometric problems

- Grid Tiling [Marx '05] $\rightarrow$ no $f(k)n^{o(\sqrt{k})}$ for several geometric packing and covering problems
- Many classical optimisation problems on multiple-interval graphs [Jiang '10, Jiang and Zhang '12]: typically W[1]-hardness on (unit) 2-track interval and (unit) 2-interval graphs

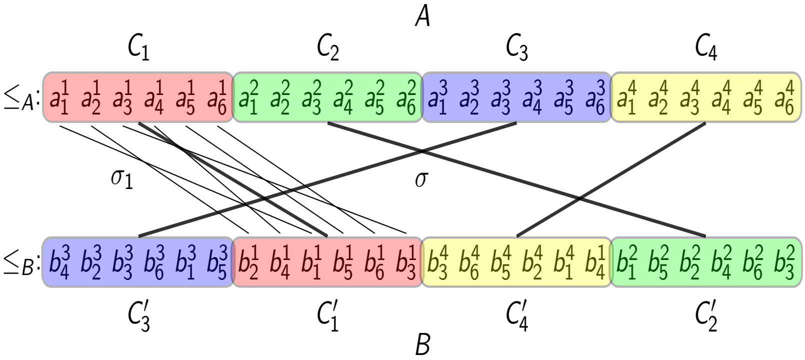
The latter can potentially give $f(k)n^{o(k/\log k)}$-lower bounds (and confirm the absence of square-root phenomenon)

Structured 2-Track Hitting Set

2-elements: $\forall i \in [t], \forall j \in [k]$ (a_i^j, b_i^j)
Total orderings of the a-elements and the b-elements
Sets: *A-intervals* and *B-intervals*
Goal: Find k 2-elements thats hits all the sets



Structured 2-Track Hitting Set



Theorem (B. & Miltzow, ESA'16)

Unless the ETH fails, Structured 2-Track Hitting Set *cannot be solved in time* $f(k)n^{o(k/\log k)}$.

Deconstructing Structured 2-Track Hitting Set

To reduce from this problem, we need to encode:

- **intervals**; usually easy
- the **interclass permutation** σ (on k elements)
- **intraclasses permutations** σ_j (on $t \gg k$ elements); trickier

Deconstructing Structured 2-Track Hitting Set

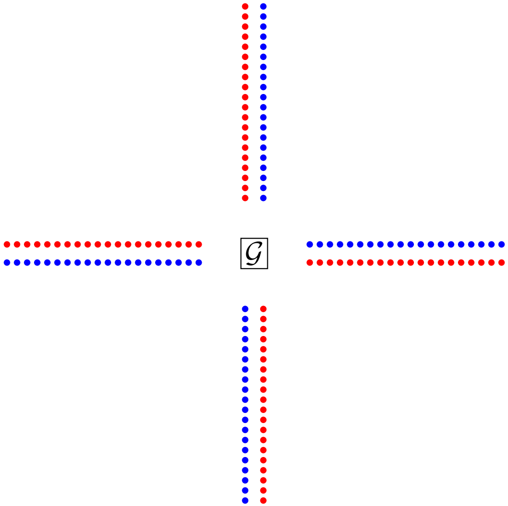
To reduce from this problem, we need to encode:

- **intervals**; usually easy
- the **interclass permutation** σ (on k elements)
- **intraclasses permutations** σ_j (on $t \gg k$ elements); trickier

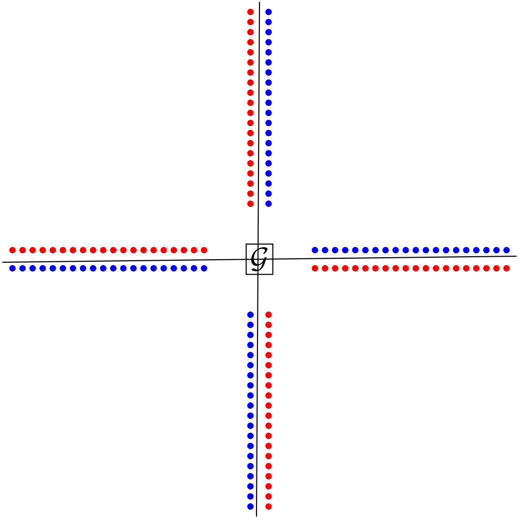
Why such an intermediate problem is convenient?

The non geometricity is pushed to **mere permutations**;
easier to simulate than
arbitrary binary relations (reduction from a graph problem) or
arbitrary ternary relations (reduction from a 3-CSP)

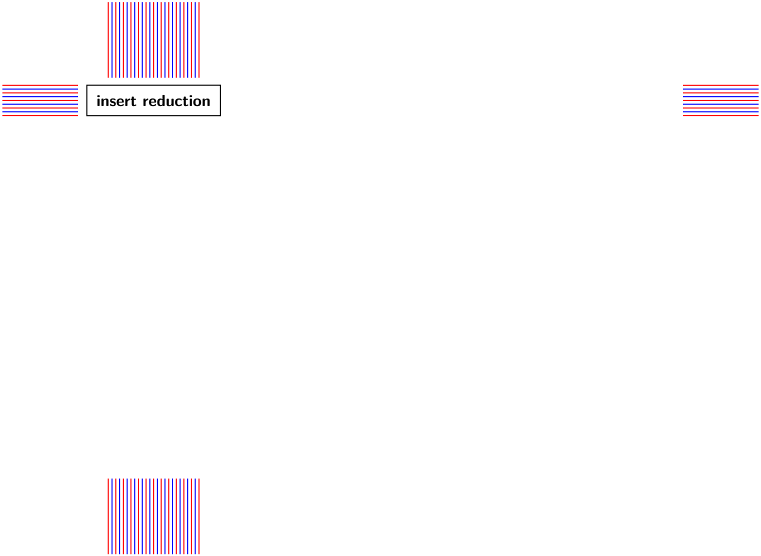
Enforcing near axis-parallelism with long alleys



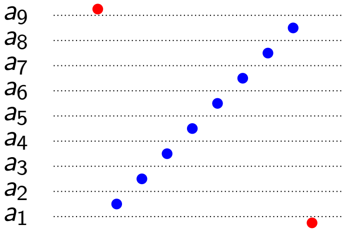
Enforcing near axis-parallelism with long alleys



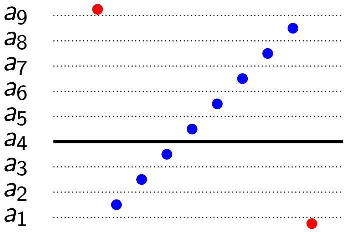
How we will in fact use them



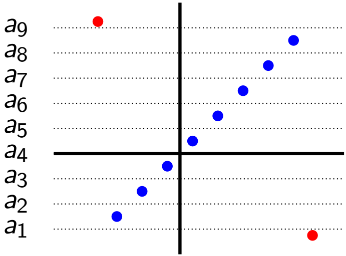
Encoding of an interval



Encoding of an interval

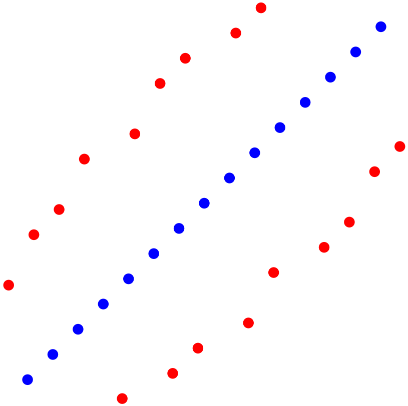


Encoding of an interval and choice propagation

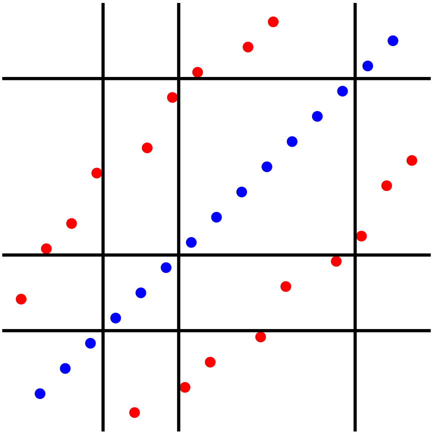


only solutions with a budget of 2 almost axis-parallel lines

Intervals put together to form the whole track

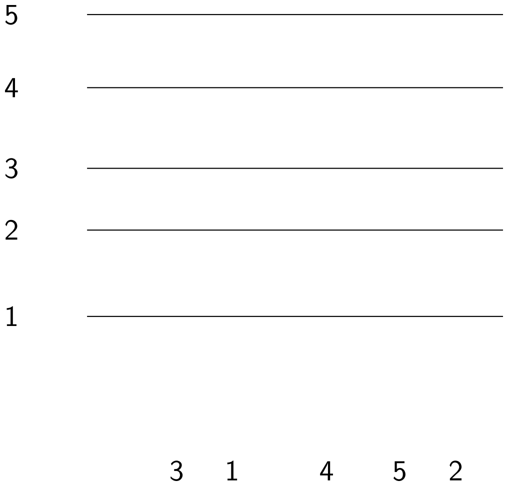


Intervals put together to form the whole track

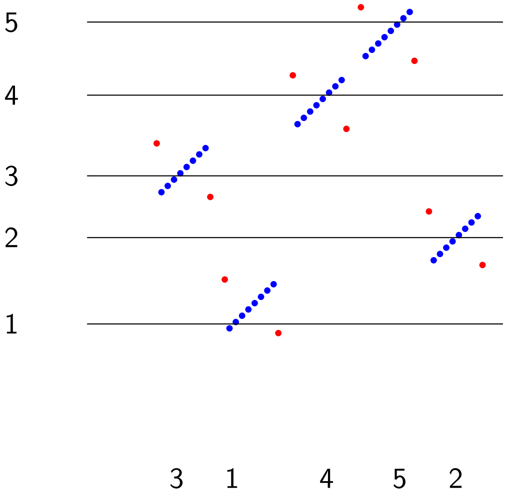


budget of k horizontal and k vertical lines

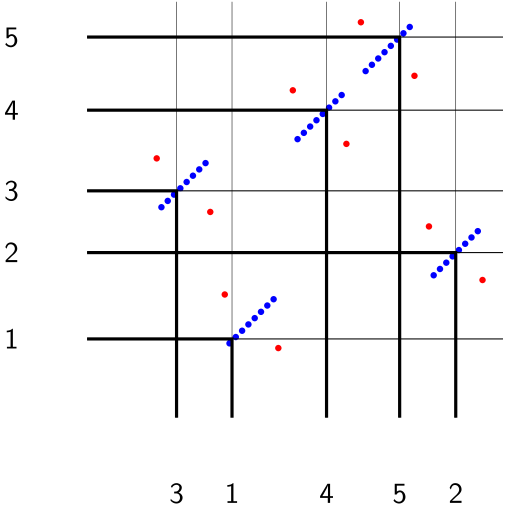
Encoding the interclass permutation σ



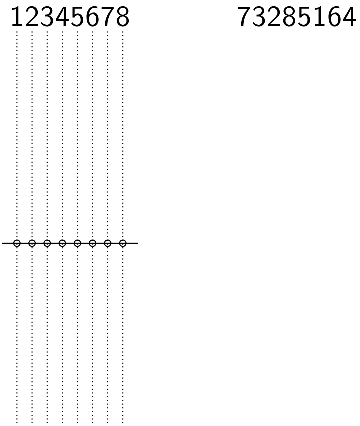
Encoding the interclass permutation σ



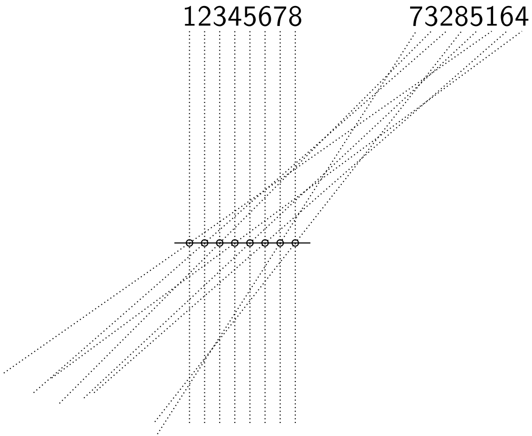
Encoding the interclass permutation σ



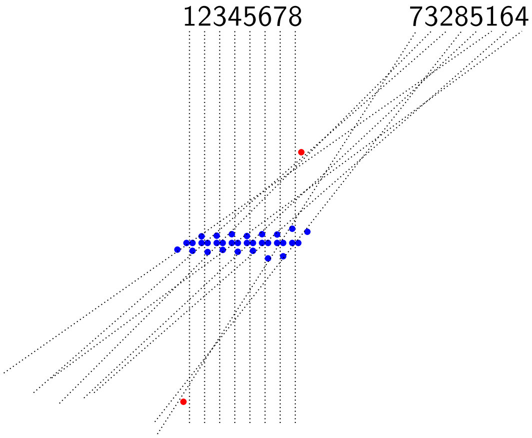
Half-encoding the intraclass permutation σ_j



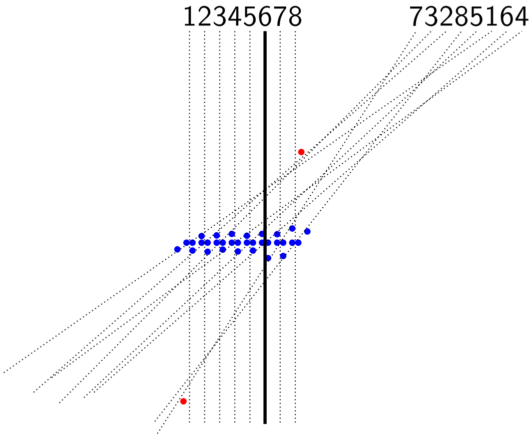
Half-encoding the intraclass permutation σ_j



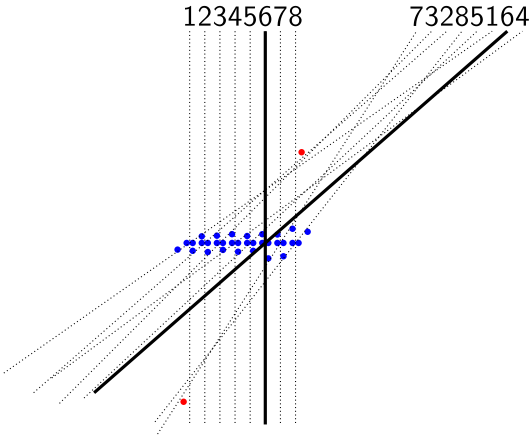
Half-encoding the intraclass permutation σ_j



Half-encoding the intraclass permutation σ_j

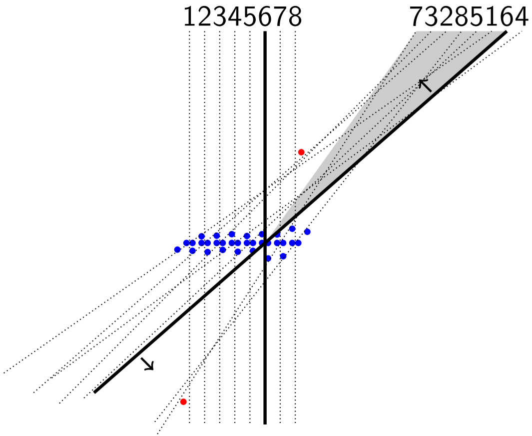


Half-encoding the intraclass permutation σ_j



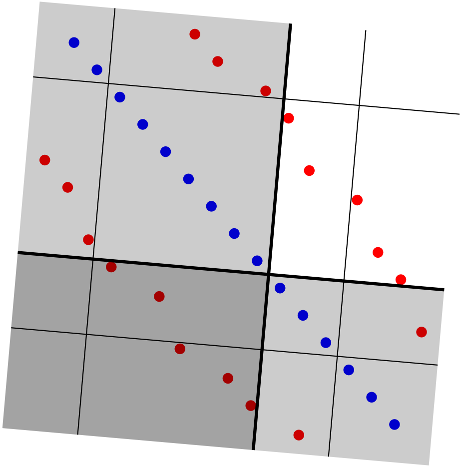
A budget of one line forces a line with the correct slope

Half-encoding the intraclass permutation σ_j



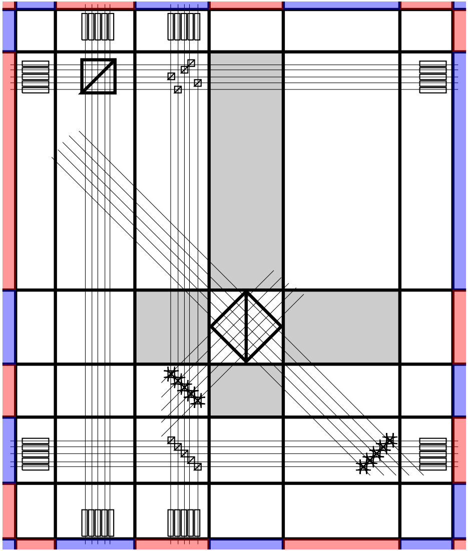
A budget of one line forces a line with the correct slope or higher

Simple fix: use two half-encodings; the second track



Gray areas correspond to possible lines; the only way to make the two lines meet at the diagonal is to take the boundary lines

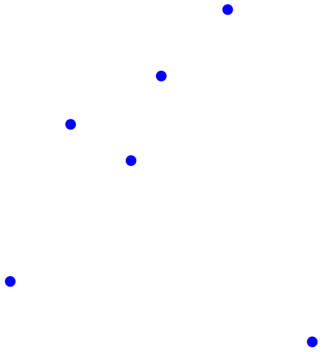
The full picture



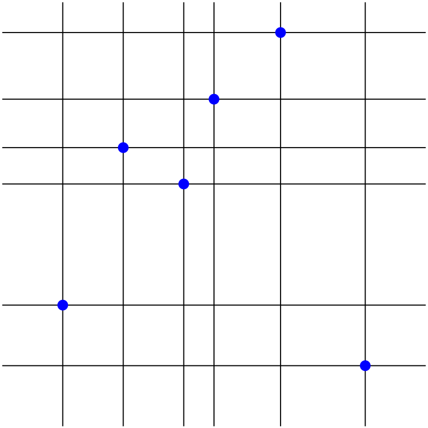
An FPT algorithm for the Axis-Parallel case[1]

Axis-Parallel Red-Blue Separtion can be solved in $O^*(9^{|\mathcal{B}|})$.

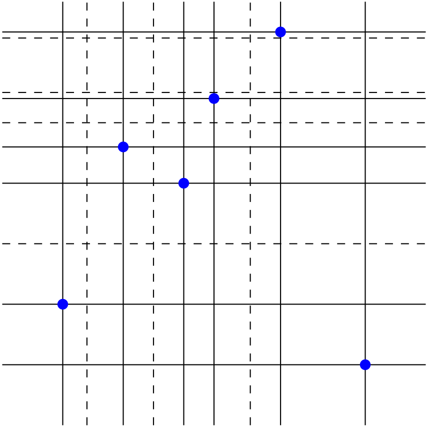
[1] with a larger parameter



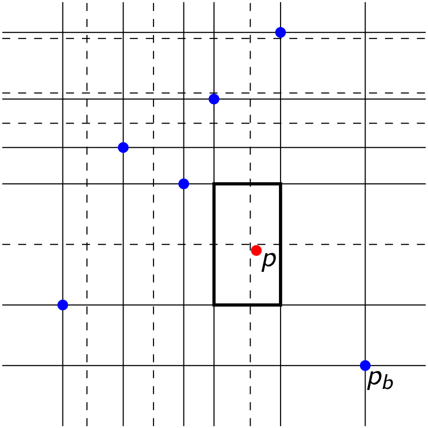
The number of blue points $k := |\mathcal{B}|$ is *small*



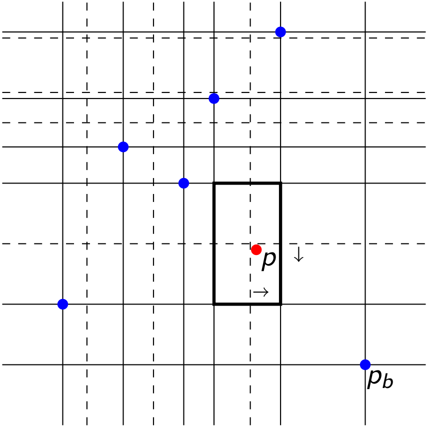
Imagine the $2k$ axis-parallel lines crossing them



Guess in time $3^{2(k+1)}$ how many lines of a solution each of the $k+1$ rows and the $k+1$ columns contain: it can be 0, 1, or 2

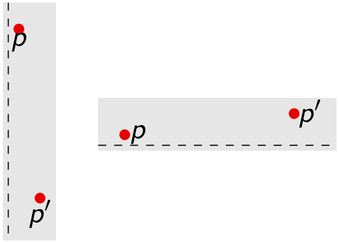


The problematic case is with 1: the lines are only *floating*



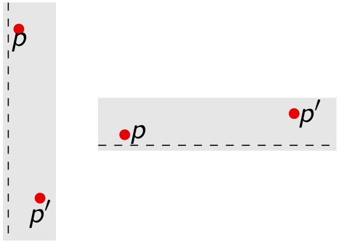
Each of the potential conflict can be expressed as a 2-clause:
VerticalLine3RightOfp or HorizontalLine1Belowp

Consistency can be ensured with 2-clauses



VerticalLineiRightOfp’ $\rightarrow$ VerticalLineiRightOfp
HorizontalLinejBelowp $\rightarrow$ HorizontalLinejBelowp’

Consistency can be ensured with 2-clauses



VerticalLineiRightOfp' $\rightarrow$ VerticalLineiRightOfp
HorizontalLinejBelowp $\rightarrow$ HorizontalLinejBelowp'

For each of the $3^{2(k+1)}$ guesses, conclude by solving a 2-SAT instance in linear time.

Open questions

The algorithm breaks with a third direction, or a third dimension, or with the natural parameter

- Is Axis-Parallel Red-Blue Separation FPT parameterized by k the number of lines?
- Is Red-Blue Separation with a fixed number of allowed slopes FPT in $|\mathcal{B}|$? in k?
- What happens in higher dimensions?

Best candidate to be FPT: Red-Blue Separation with 3 allowed slopes and parameterized by $|\mathcal{B}|$.