Optimal Discretization is Fixed-Parameter Tractable

Stefan Kratsch

HU Berlin

Tomáš Masařík

U Warsaw

→ SFU

Irene Muzi

U Warsaw

→ TU Berlin

Marcin Pilipczuk

U Warsaw

Manuel Sorge

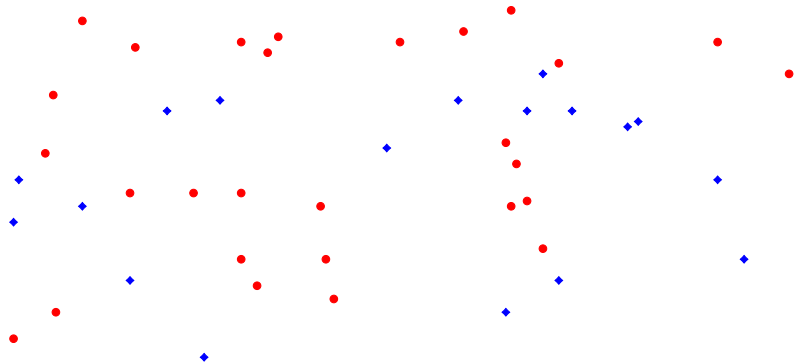
U Warsaw

→ TU Wien

Simon Fraser University Discrete Seminar 2021

Optimal Discretization

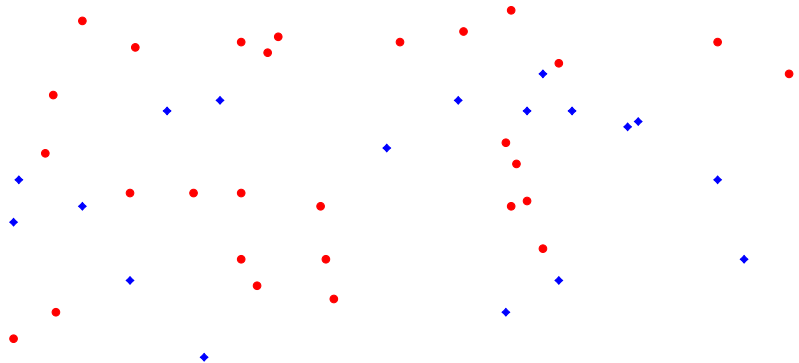
Input: Two disjoint sets Red and Blue of points in $\mathbb{R}^2$ and an integer k.



Optimal Discretization

Input: Two disjoint sets Red and Blue of points in $\mathbb{R}^2$ and an integer k.

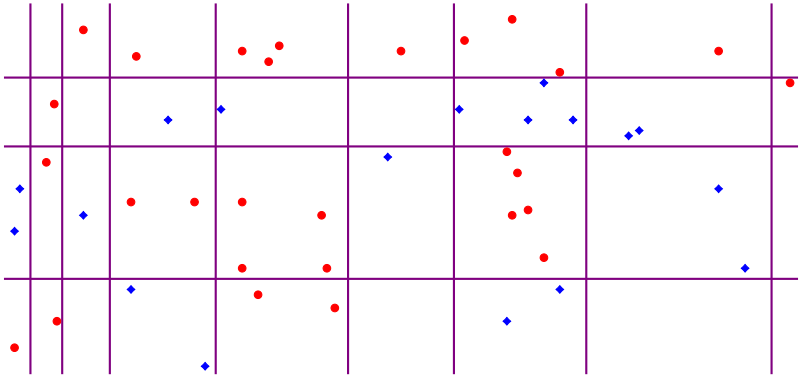
Goal: A set of k horizontal/vertical lines separating Red from Blue.



Optimal Discretization

Input: Two disjoint sets Red and Blue of points in $\mathbb{R}^2$ and an integer k.

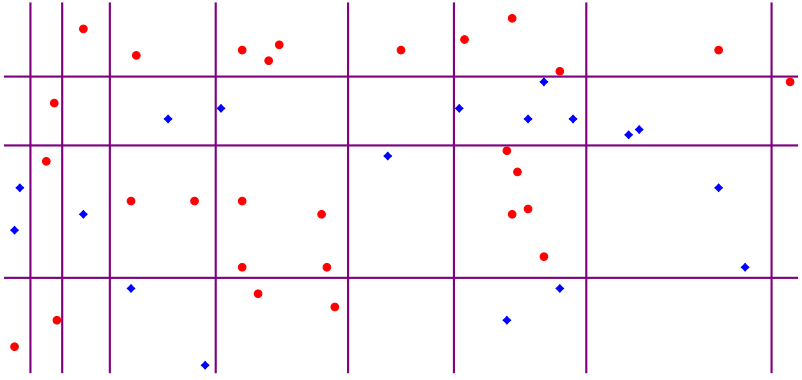
Goal: A set of k horizontal/vertical lines separating Red from Blue.



Optimal Discretization

Input: Two disjoint sets Red and Blue of points in $\mathbb{R}^2$ and an integer k.

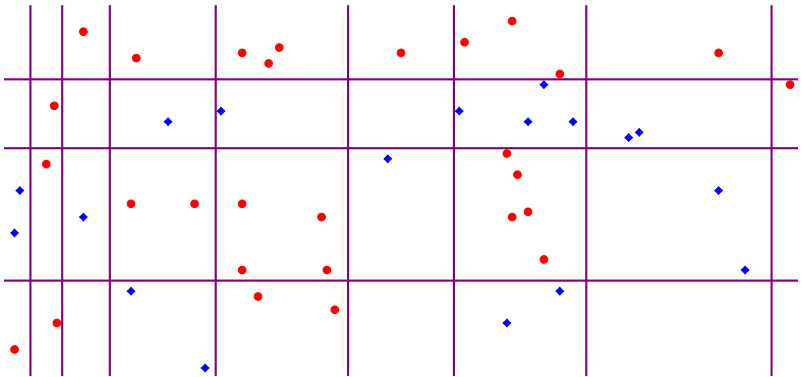
Goal: A set of k horizontal/vertical lines separating Red from Blue.



NP-hard! [Megiddo, Discrete Comput. Geom. 1988]

Machine Learning motivation: round coordinates to as few as possible values
[Witten & Frank, Data Mining; Froese, PhD thesis] without merging points from Red and Blue

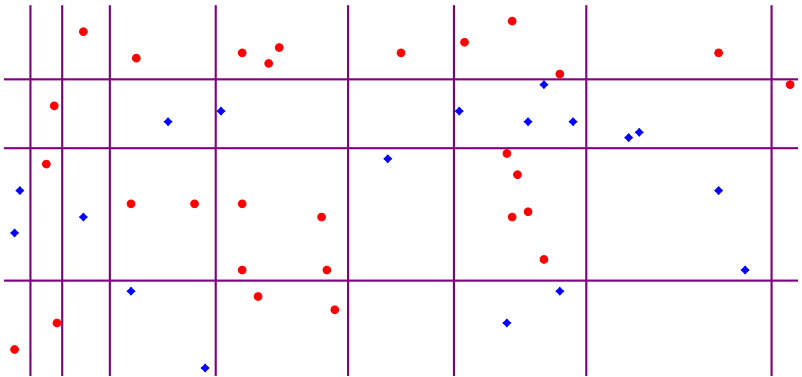
Parameterized context on Optimal Discretization



- **FPT** when parameterized by min(|Red|, |Blue|).
- W[1]-hard when parameterized by k if lines with arbitrary slopes allowed.

[Bonnet, Giannopoulos, Lampis, IPEC 2017]

Parameterized context on Optimal Discretization

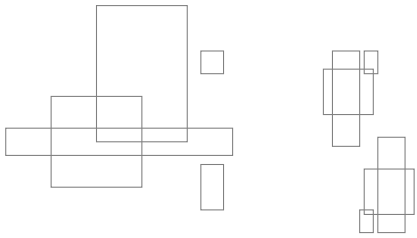


- $f(k)n^{O(1)}$ when parameterized by $k := \min(|\text{Red}|, |\text{Blue}|)$.
- W[1]-hard when parameterized by k if lines with arbitrary slopes allowed.

[Bonnet, Giannopoulos, Lampis, IPEC 2017]

Rectangle Stabbing

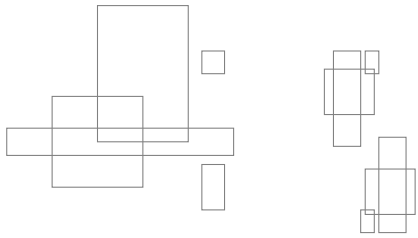
Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.



Rectangle Stabbing

Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.

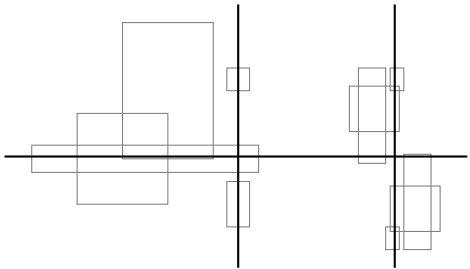
Goal: A set of k horizontal/vertical lines intersecting every rectangle.



Rectangle Stabbing

Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.

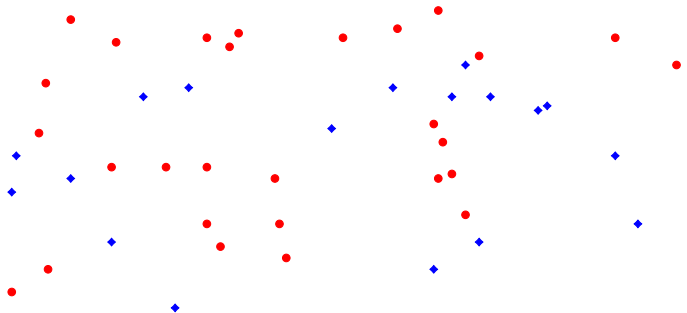
Goal: A set of k horizontal/vertical lines intersecting every rectangle.



Rectangle Stabbing

Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.

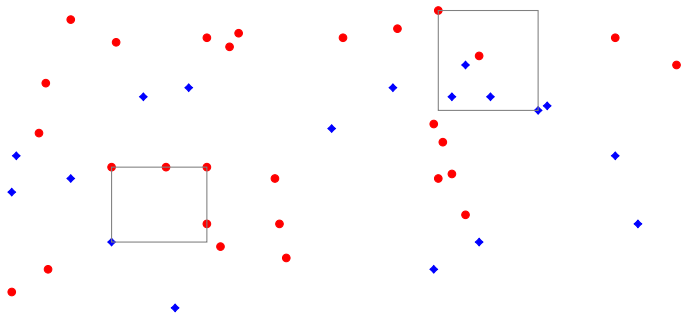
Goal: A set of k horizontal/vertical lines intersecting every rectangle.



Rectangle Stabbing

Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.

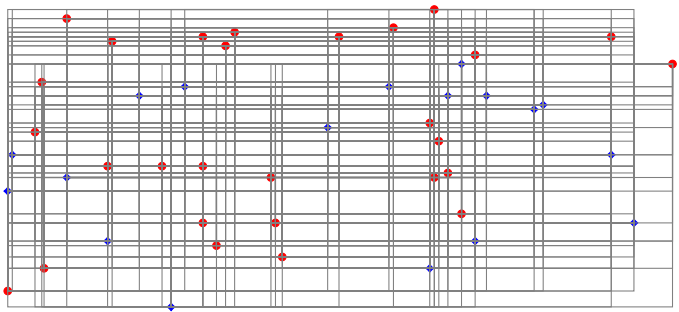
Goal: A set of k horizontal/vertical lines intersecting every rectangle.



Rectangle Stabbing

Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.

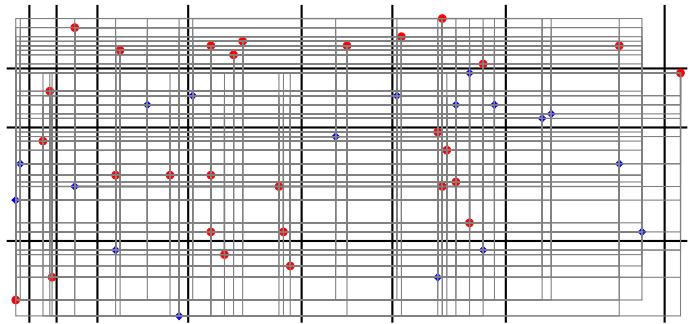
Goal: A set of k horizontal/vertical lines intersecting every rectangle.



Rectangle Stabbing

Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.

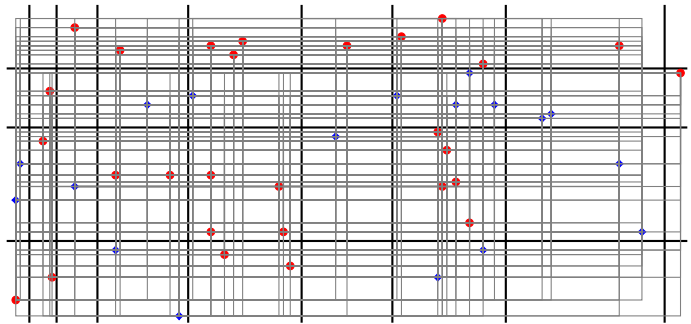
Goal: A set of k horizontal/vertical lines intersecting every rectangle.



Rectangle Stabbing

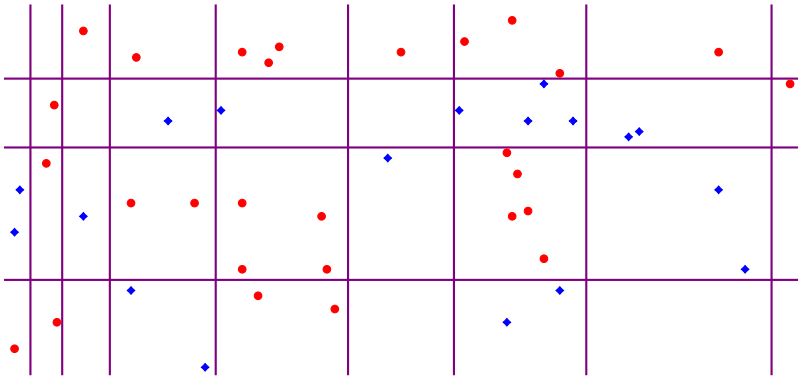
Input: A set of axis-parallel rectangles in $\mathbb{R}^2$ and an integer k.

Goal: A set of k horizontal/vertical lines intersecting every rectangle.



- Rectangle Stabbing is W[1]-hard when parameterized by k.
  [Dom, Fellows, Rosamond, Sikdar., Algorithmica 2012]
- Disjoint Rectangle Stabbing is fixed-parameter tractable when parameterized by k.
  [Heggernes, Kratsch, Lokshtanov, Raman, Saurabh, Information and Computation 2013]

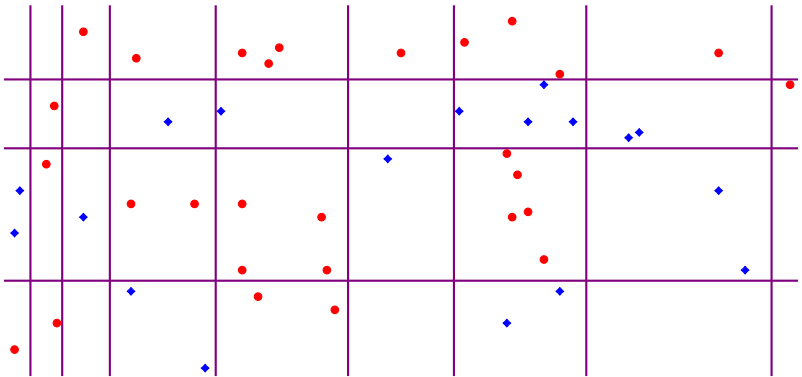
Optimal Discretization



Theorem (SK, TM, IM, MP, MS)

Optimal Discretization *can be solved in* $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$ *time.*

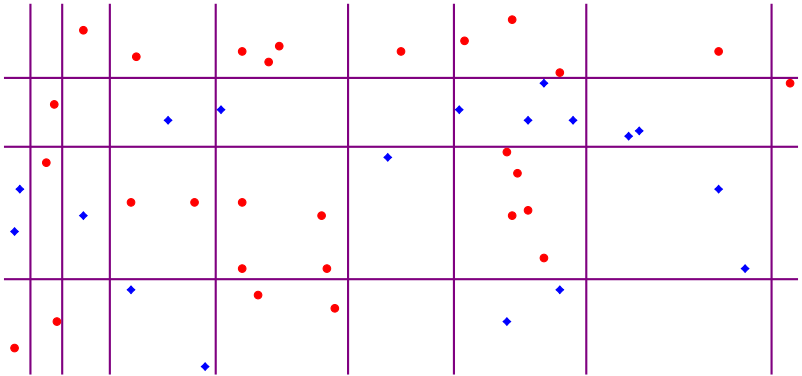
Start: 2-approximate solution



[Călinescu, Dumitrescu, Karloff, Wan. Int. J. Comput. Geometry Appl., 2005]

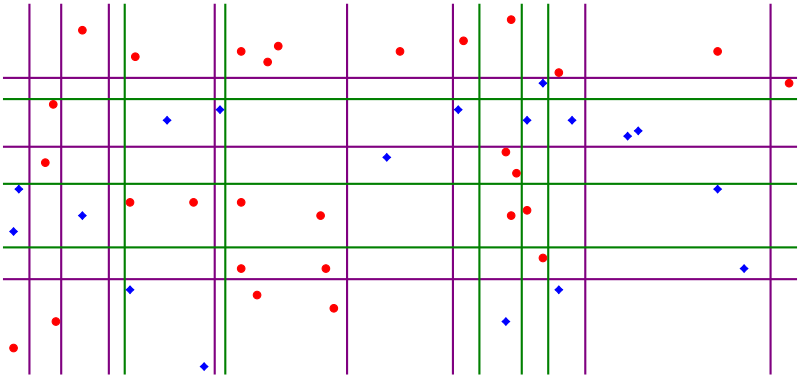
Old and new solutions

Given an instance and a **old solution** of size at most $11 \leq 2k$, find a **new solution** of size k.



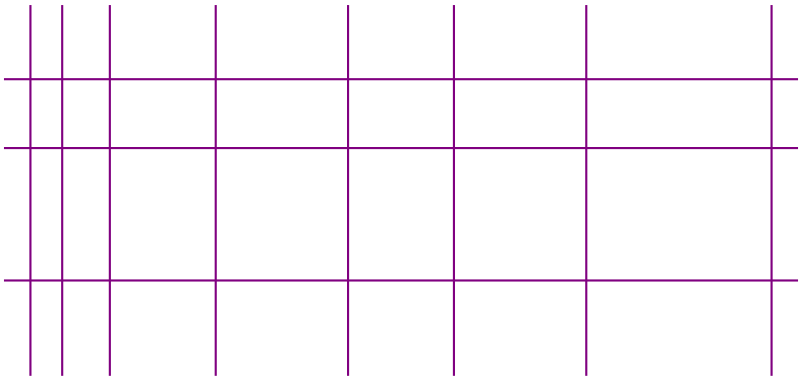
Old and new solutions

Given an instance and a **old solution** of size at most $11 \leq 2k$, find a **new solution** of size $k = 9$.



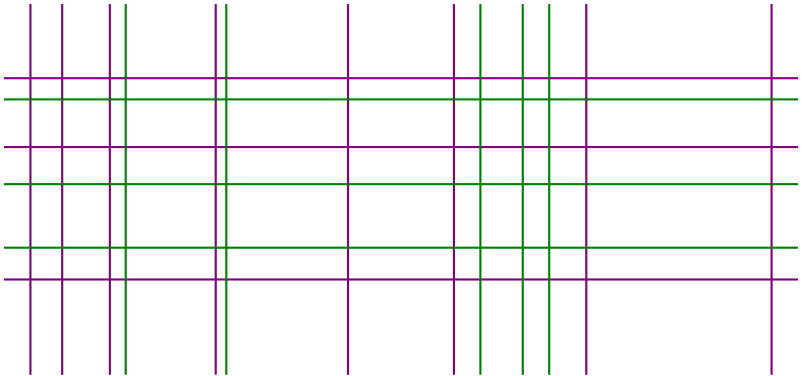
Branching I.—Guess layout of k lines

Given an instance and a **solution** of size at most $2k$, find a **solution** of size $k = 9$.
In total $(4k + 2)^k$ branches.



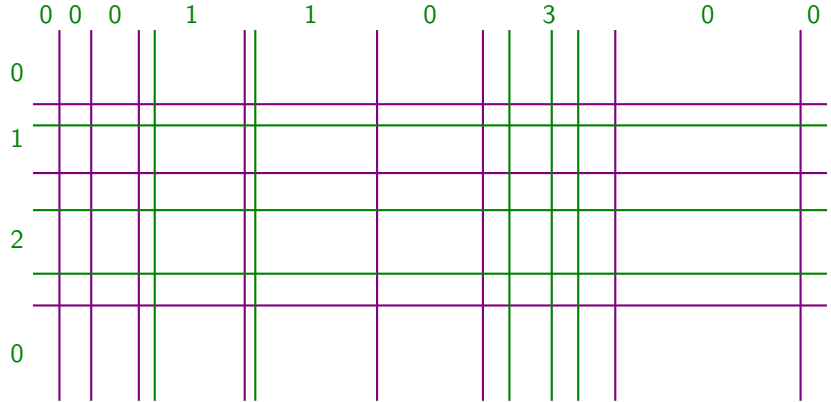
Branching I.—Guess layout of k lines

Given an instance and a **solution** of size at most $2k$, find a **solution** of size $k = 9$.
In total $(4k + 2)^k$ branches.



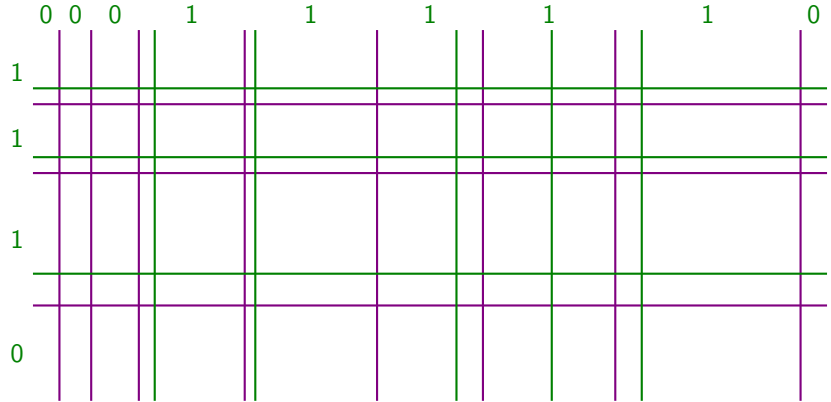
Branching I.—Guess layout of k lines

Given an instance and a **solution** of size at most $2k$, find a **solution** of size $k = 9$.
In total $(4k + 2)^k$ branches.



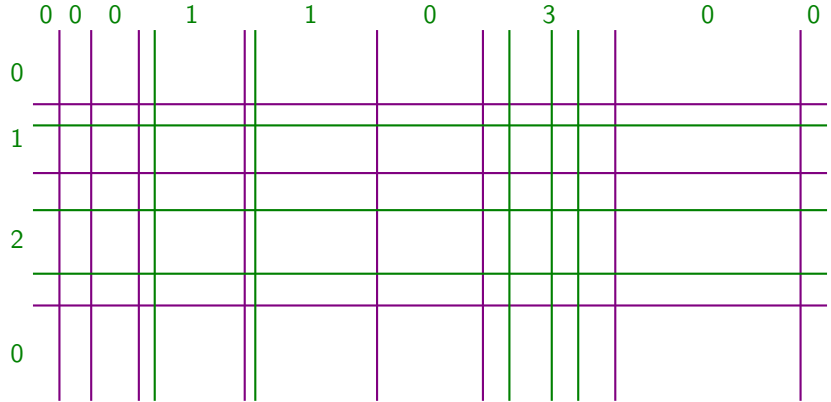
Branching I.—Guess layout of k lines

Given an instance and a **solution** of size at most $2k$, find a **solution** of size $k = 9$.
In total $(4k + 2)^k$ branches. Simple case.



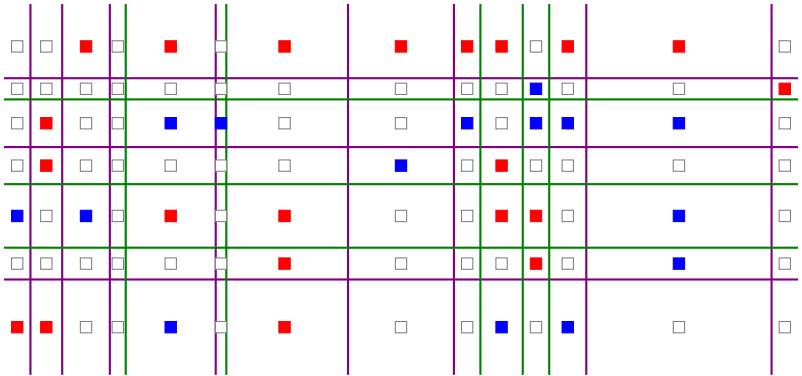
Branching II.—Guess which cells are empty?

Branch for each cell whether it is empty or not.
In total $2^{O(k^2)}$ branches.



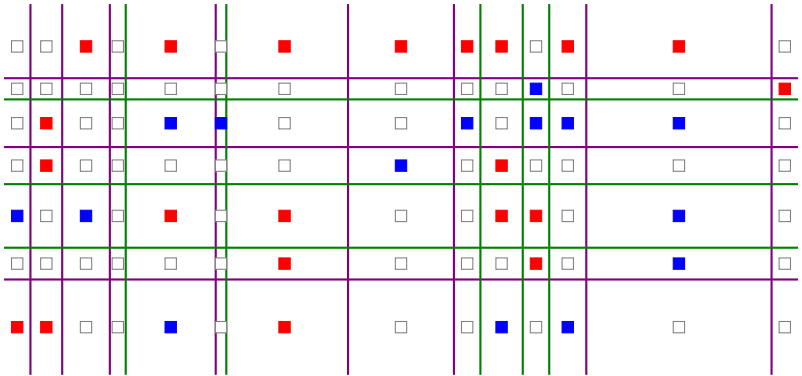
Branching II.—Guess which cells are empty?

Branch for each cell whether it is empty or not.
In total $2^{O(k^2)}$ branches.



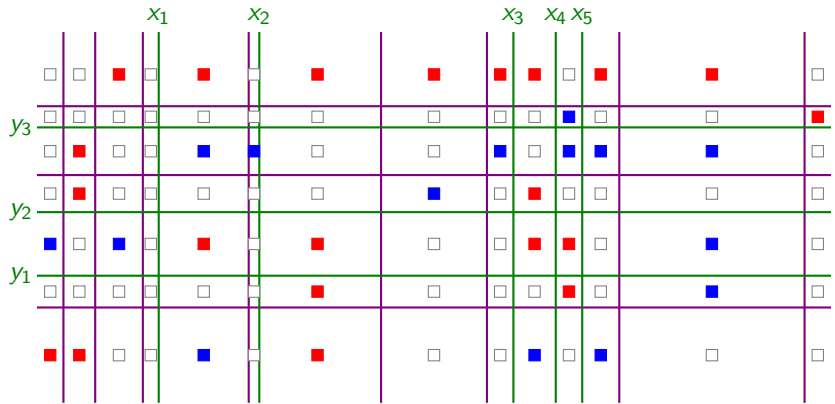
Branching II.—Guess which cells are empty?

Branch for each cell whether it is empty or not.
In total $2^{O(k^2)}$ branches.

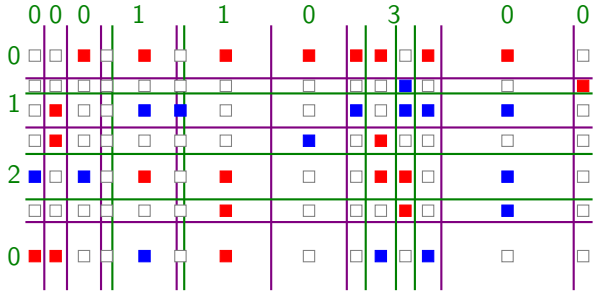


Branching II.—Guess which cells are empty?

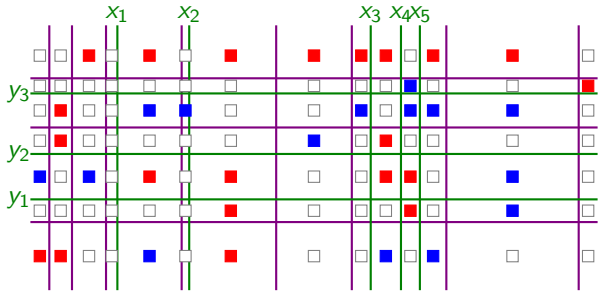
Branch for each cell whether it is empty or not.
In total $2^{O(k^2)}$ branches.



CSP (Constraints Satisfaction Problem) formulation



CSP (Constraints Satisfaction Problem) formulation



Variables: Coordinates x_i of green vertical lines, y_i for green horizontal lines.

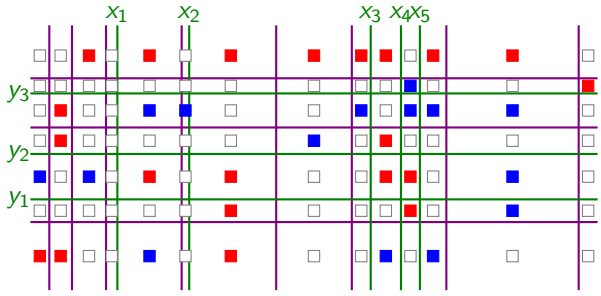
Domains: $\mathrm{Dom}(x_i) :=$ possible coordinates between purple lines next to x_i; same for y_i.

Constraints: Monotonicity: $x_i < x_j$ and $y_i < y_j$ for every $i < j$ between the same purple lines.

Consistency: $\forall p \in$ Red and $\forall$ non-red cell C : $p \notin C$.

$\forall p \in$ Blue and $\forall$ non-blue cell C : $p \notin C$.

CSP (Constraints Satisfaction Problem) formulation



Variables: Coordinates x_i of green vertical lines, y_i for green horizontal lines.

Domains: $\mathrm{Dom}(x_i) :=$ possible coordinates between purple lines next to x_i; same for y_i.

Constraints: Monotonicity: $x_i < x_j$ and $y_i < y_j$ for every $i < j$ between the same purple lines.

Consistency: $\forall p \in$ Red and $\forall$ non-red cell C : $p \notin C$.

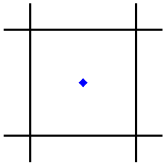
$\forall p \in$ Blue and $\forall$ non-blue cell C : $p \notin C$.

The above CSP instance is equivalent to the input Red-Blue Points with guessed overlay.

How complicated the constraints are?

Constraint: 0 and 1 green lines

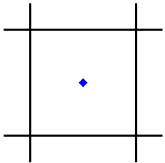
Case 0: 0 green lines



Point $p = (x^p, y^p) \notin C$:
always true or always false.
Reject overlay guess if false.

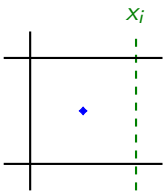
Constraint: 0 and 1 green lines

Case 0: 0 green lines



Point $p = (x^p, y^p) \notin C$:
always true or always false.
Reject overlay guess if false.

Case 1: 1 green line



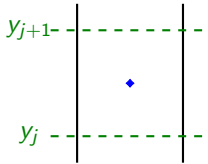
Point $p = (x^p, y^p) \notin C$:

$$(x_i < x^p).$$

We can restrict the domain of x_i.

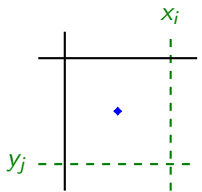
Constraint: 2 green lines

Case 2: 2 green lines



Point $p = (x^p, y^p) \notin C$:

$$(y_j > y^p) \vee (y_{j+1} < y^p).$$

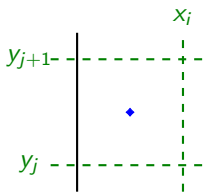


Point $p = (x^p, y^p) \notin C$:

$$(x_i < x^p) \vee (y_j > y^p).$$

Constraint: 3 and 4 green lines

Case 3: 3 green lines

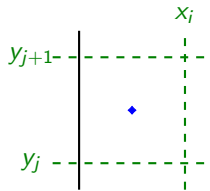


Point $p = (x^p, y^p) \notin C$:

$$(x_i < x^p) \vee (y_j > y^p) \vee (y_{j+1} < y^p).$$

Constraint: 3 and 4 green lines

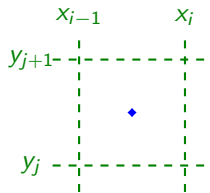
Case 3: 3 green lines



Point $p = (x^p, y^p) \notin C$:

$$(x_i < x^p) \vee (y_j > y^p) \vee (y_{j+1} < y^p).$$

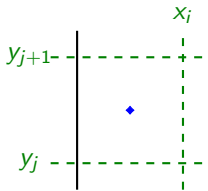
Case 4: 4 green lines



Point $p = (x^p, y^p) \notin C$:

Constraint: 3 and 4 green lines

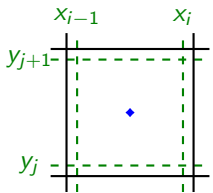
Case 3: 3 green lines



Point $p = (x^p, y^p) \notin C$:

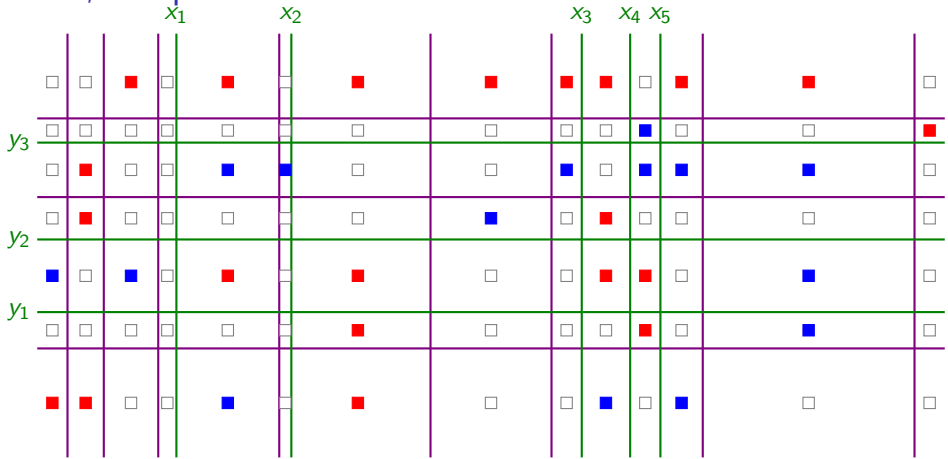
$$(x_i < x^p) \vee (y_j > y^p) \vee (y_{j+1} < y^p).$$

Case 4: 4 green lines



Point $p = (x^p, y^p) \notin C$:
We can ignore, cell always monochromatic.

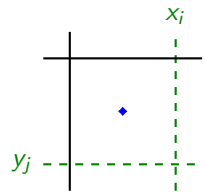
Constraints, recap



Constraints, recap

Monotonicity: $x_i < x_{i+1}$, $y_j < y_{j+1}$.

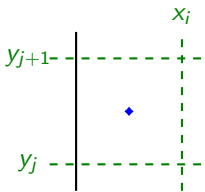
Case 2 cells:



Point $p = (x^p, y^p) \notin C$:

$$(x_i < x^p) \vee (y_j > y^p).$$

Case 3 cells:



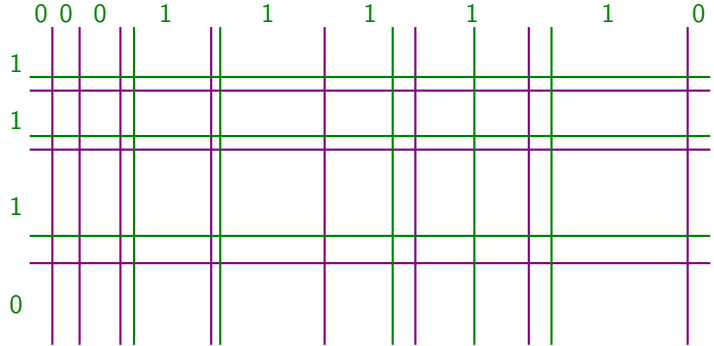
Point $p = (x^p, y^p) \notin C$:

$$(x_i < x^p) \vee (y_j > y^p) \vee (y_{j+1} < y^p).$$

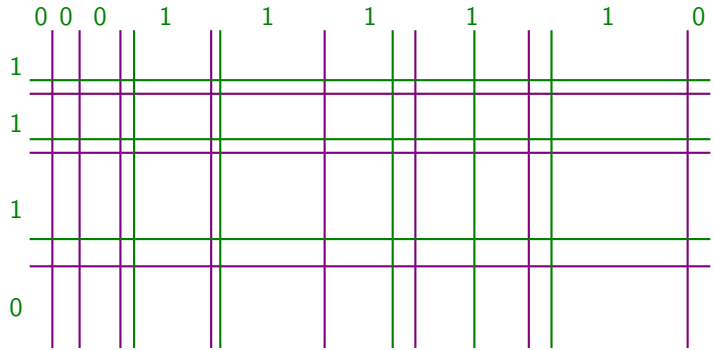
Important special case

Simple case—At most one new line between two old lines

Observe no Case 3 cells.



Important special case



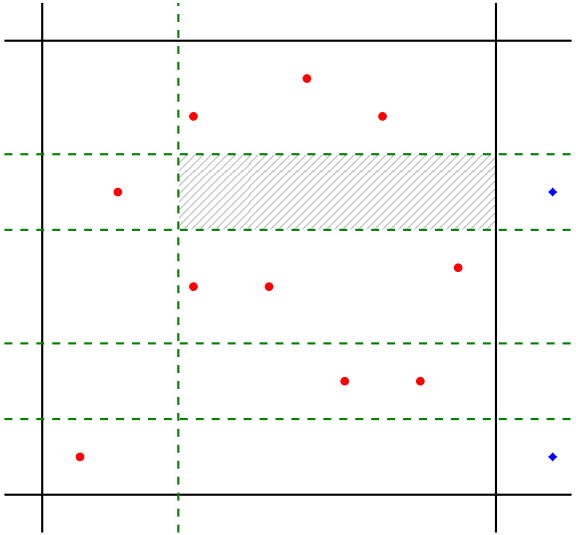
Theorem

Such a CSP is polynomial-time solvable.

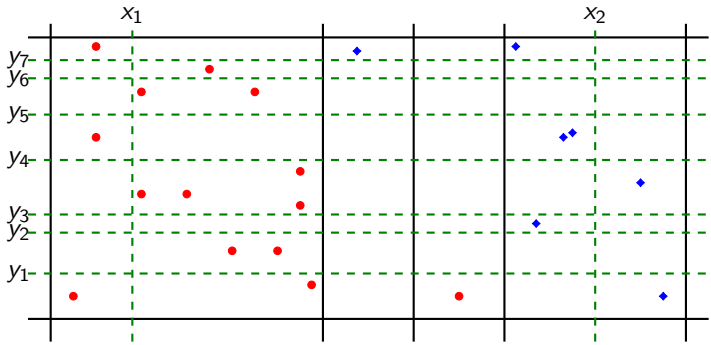
Proof via CSP: median is a majority polymorphism.

Alternative proof: reduce to 2-CNF SAT (easy).

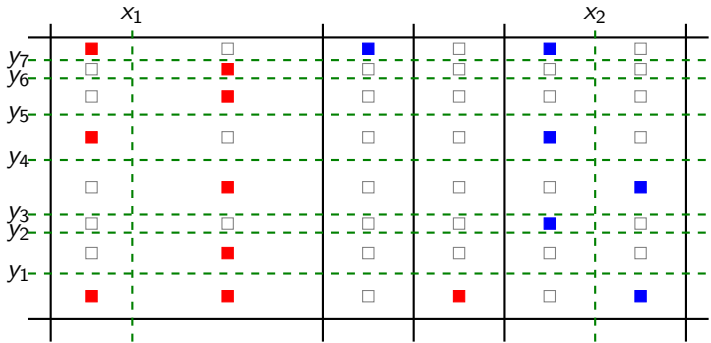
Case 3 constraint is important



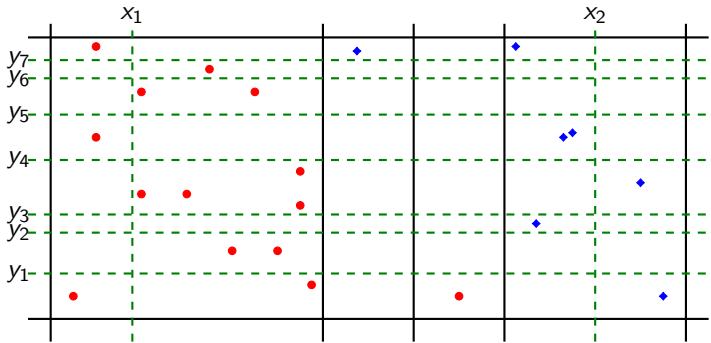
Case 3



Case 3



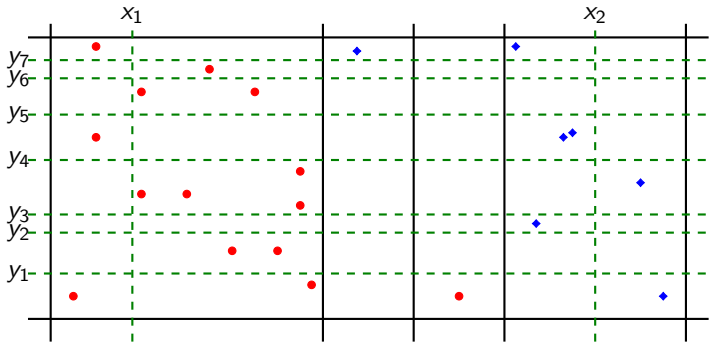
Case 3



Observation: For a fixed value of x_1 (regardless of the value of x_2),

- the set of Red points is fixed.
- there is a segment of values of x_2 that gives the guessed pattern.

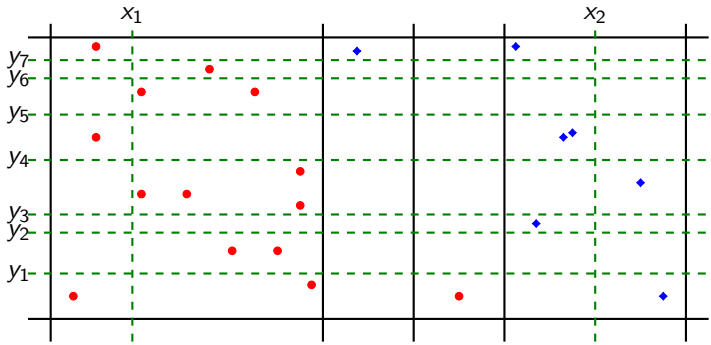
Case 3



Observation: For a fixed value of x_1 (regardless of the value of x_2),

- the set of Red points is fixed.
- there is a segment of values of x_2 that gives the guessed pattern.
- all Red points are partitioned into blocks in the same way.

Case 3

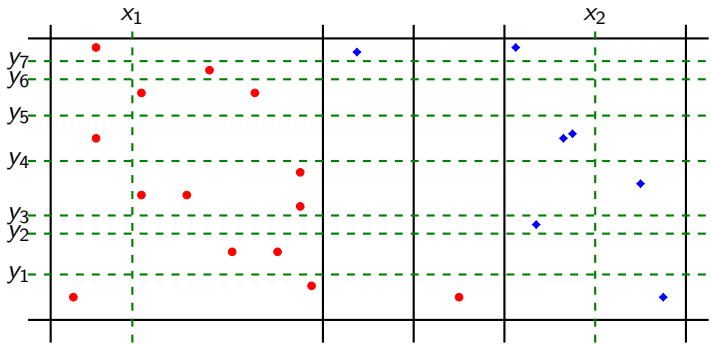


Observation: For a fixed value of x_1 (regardless of the value of x_2),

- the set of Red points is fixed.
- there is a segment of values of x_2 that gives the guessed pattern.
- all Red points are partitioned into blocks in the same way.

Constraint: For a fixed value of x_1, allow only values of y_i in the correct position between the blocks.

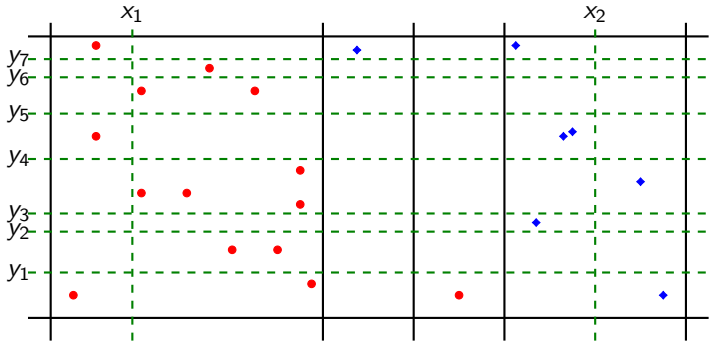
Case 3 reengineering



In every such picture:

- Bind x_1 and x_2 with a constraint "the pattern is as guessed".
- Bind x_1 and y_i with a constraint "y_i is between the correct blocks of points".

Case 3 reengineering

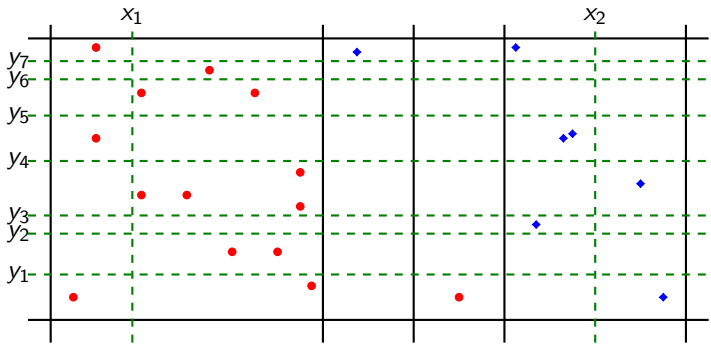


In every such picture:

- Bind x_1 and x_2 with a constraint "the pattern is as guessed".
- Bind x_1 and y_i with a constraint "y_i is between the correct blocks of points".

Claim: This is equivalent to Case 3 constraints.

Case 3 reengineering



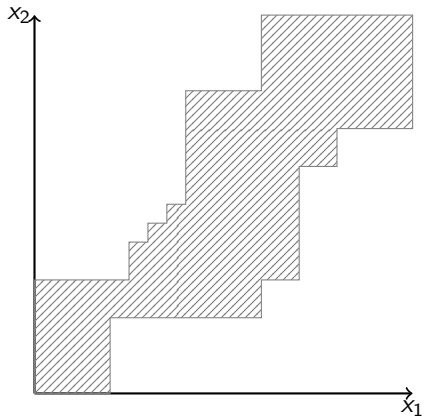
In every such picture:

- Bind x_1 and x_2 with a constraint "the pattern is as guessed".
- Bind x_1 and y_i with a constraint "y_i is between the correct blocks of points".

Claim: This is equivalent to Case 3 constraints.

Keep the pattern constraint

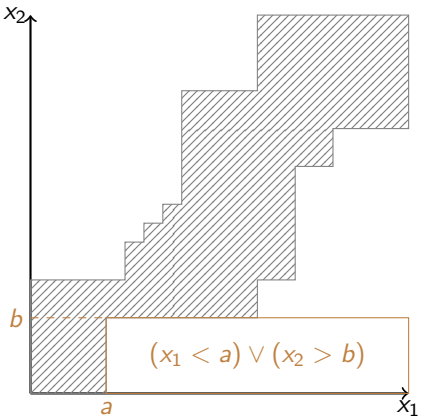
Bind x_1 and x_2 with a constraint "the pattern is as guessed".



Conjunction of a number of constaints
$(x_1 < a) \vee (x_2 > b)$ and $(x_1 > a) \vee (x_2 < b)$.

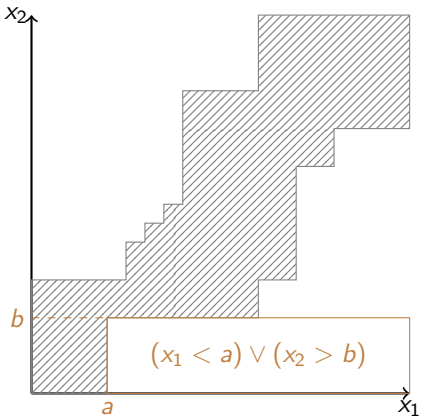
Keep the pattern constraint

Bind x_1 and x_2 with a constraint "the pattern is as guessed".



Keep the pattern constraint

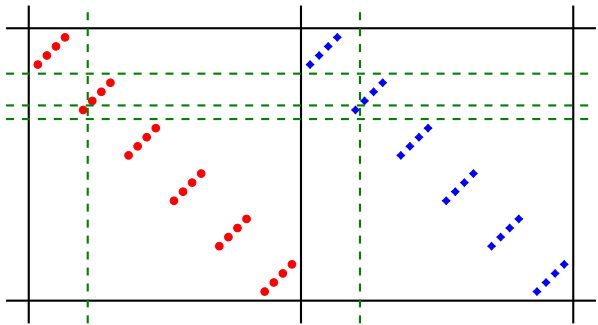
Bind x_1 and x_2 with a constraint "the pattern is as guessed".



Conjunction of a number of constaints
$(x_1 < a) \vee (x_2 > b)$ and $(x_1 > a) \vee (x_2 < b)$.

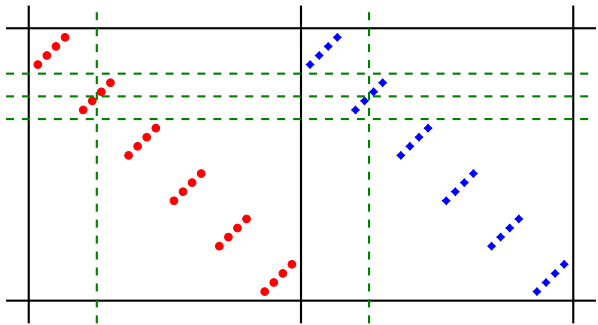
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



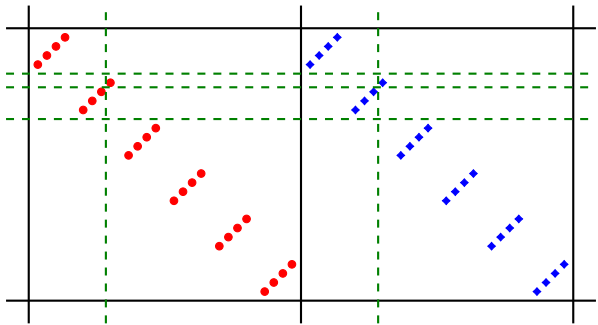
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



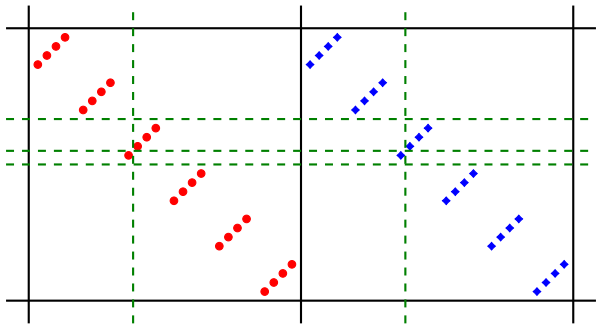
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



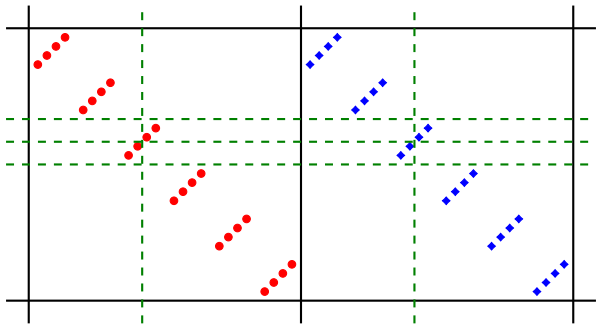
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



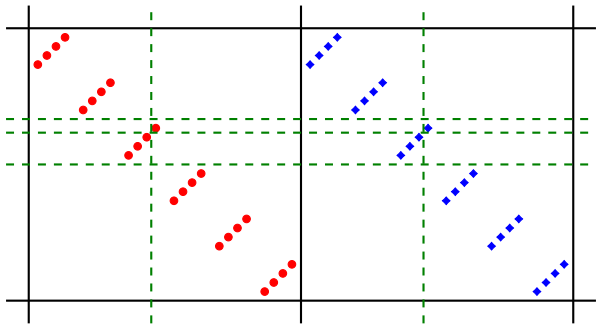
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



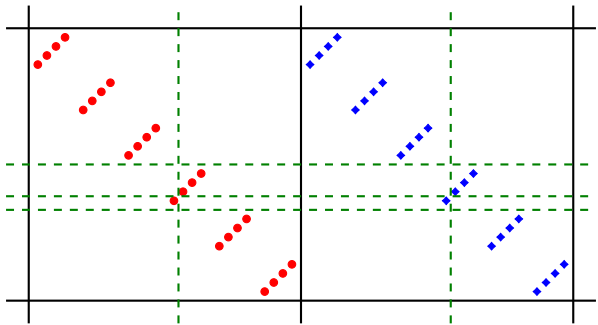
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



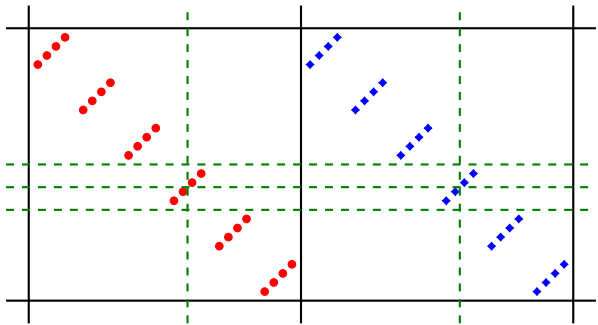
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



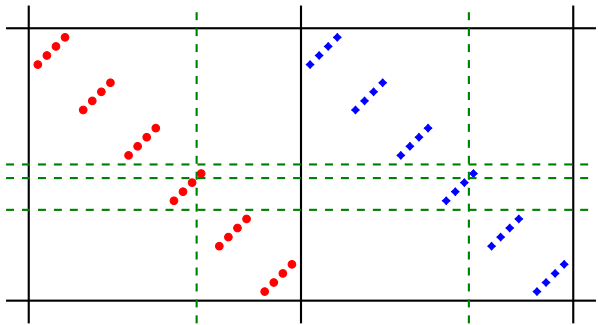
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



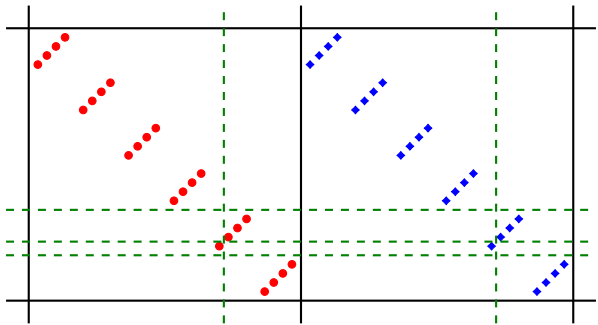
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



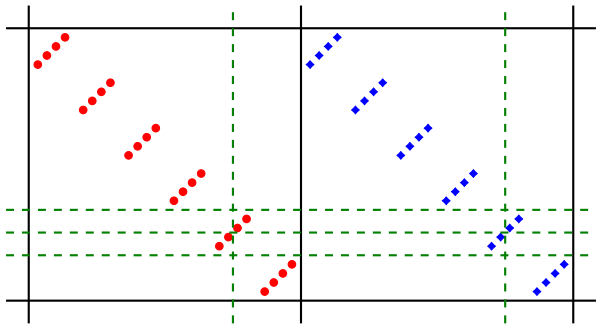
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



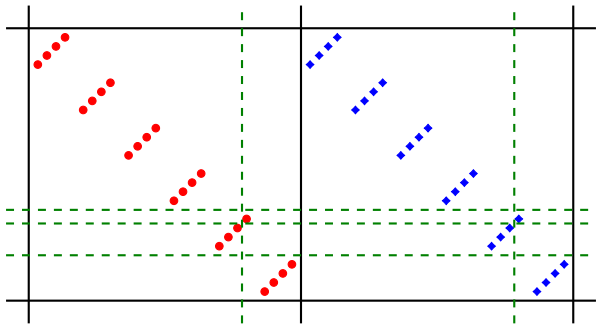
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



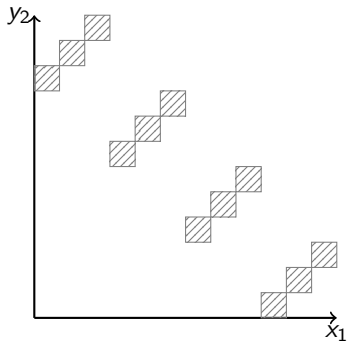
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



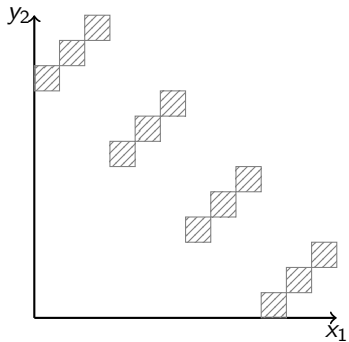
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



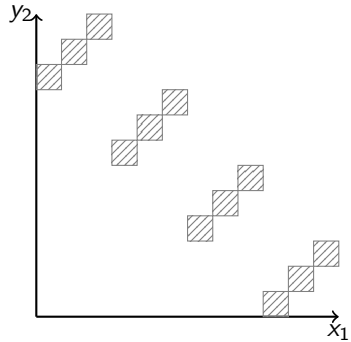
Separate blocks constraint

Bind x_1 and y_i with a constraint
"y_i is between the correct blocks of points".



Not expressible as previously.

Idea



- Stop playing with Red-Blue Points.
- Develop a more general CSP language: (Auxiliary Forest CSP)
  - Still FPT parameterized by the number of variables.
  - Captures constaints as above.
- Then go back to Red-Blue Points and try to reduce.

Wrap up

Theorem (SK, TM, IM, MP, MS)

Auxiliary Forest CSP *with p variables can be solved in $2^{\mathcal{O}(p \log p)} n^{\mathcal{O}(1)}$ time.*

A lot of branching and color coding steps $\rightarrow$
encoding of Red-Blue Points as Auxiliary Forest CSP with $\mathcal{O}(k^2)$ variables.

Theorem (SK, TM, IM, MP, MS)

Optimal Discretization *can be solved in time $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$.*

Crucial: the extra power of Auxiliary Forest CSP seems needed.

Projecting the branching algorithm back to Red-Blue Points is super-cumbersome.

Conclusions

Theorem (SK, TM, IM, MP, MS)

Optimal Discretization *can be solved in time* $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$.

Theorem (SK, TM, IM, MP, MS)

Auxiliary Forest CSP *can be solved in time* $2^{\mathcal{O}(p \log p)} n^{\mathcal{O}(1)}$.

Conclusions

Theorem (SK, TM, IM, MP, MS)

Optimal Discretization *can be solved in time* $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$.

Theorem (SK, TM, IM, MP, MS)

Auxiliary Forest CSP *can be solved in time* $2^{\mathcal{O}(p \log p)} n^{\mathcal{O}(1)}$.

Open problem:
Theory of parameterized complexity of CSPs parameterized by the number of variables?

Conclusions

Theorem (SK, TM, IM, MP, MS)

Optimal Discretization *can be solved in time* $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(1)}$.

Theorem (SK, TM, IM, MP, MS)

Auxiliary Forest CSP *can be solved in time* $2^{\mathcal{O}(p \log p)} n^{\mathcal{O}(1)}$.

Open problem:
Theory of parameterized complexity of CSPs parameterized by the number of variables?

Thank you!