Fast Fencing

Mikkel Abrahamsen
University of Copenhagen

Anna Adamaszek
University of Copenhagen

Karl Bringmann
Max Planck Institute

Vincent Cohen-Addad
CNRS & Sorbonne Université

Mehran Mehr
TU Eindhoven

Eva Rotenberg
TU of Denmark

Alan Roytman
University of Copenhagen

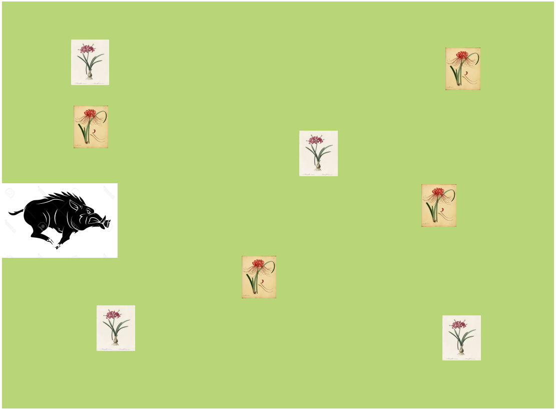
Mikkel Thorup
University of Copenhagen

J. Gwyn Delin. — Hall Sculp.

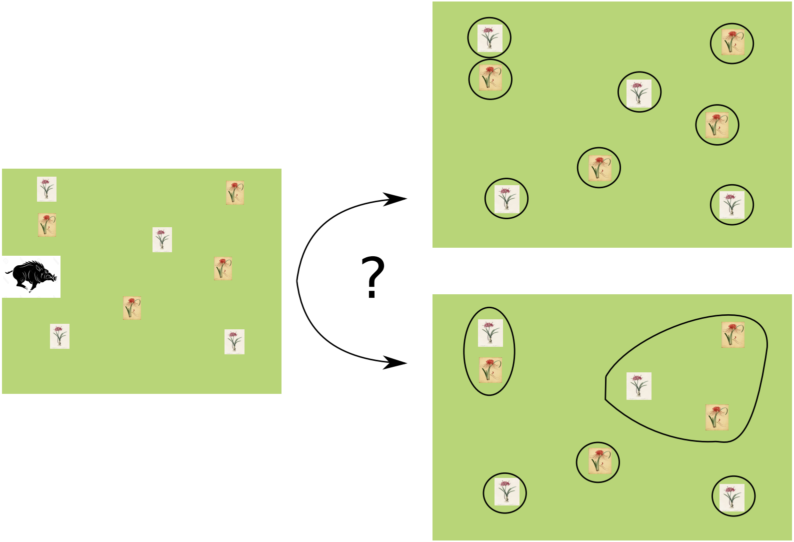
Demi volte sur les coups forcés au dehors des armes. Plate 32.

Publish'd according to Act of Parliament, Feb. 1763.

How to protect these flowers from wild animals?



How to protect these flowers from wild animals?



Which one of the two is the least expensive in fences?

More formally

Unit-disk fencing problem

Input: A set D of unit-disks in $\mathbb{R}^2$.
Output: A partition $\{D_1, \ldots, D_\ell\}$ of D that minimizes

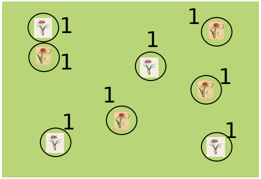
$$\sum_{i=1}^{\ell} |\mathrm{ConvexHull}(D_i)|.$$

More formally

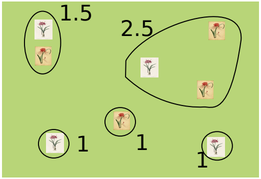
Unit-disk fencing problem

Input: A set D of unit-disks in $\mathbb{R}^2$.
Output: A partition $\{D_1, \ldots, D_\ell\}$ of D that minimizes

$$\sum_{i=1}^{\ell} |\text{ConvexHull}(D_i)|.$$



total cost = 8



total cost = 7

More formally

Unit-disk fencing problem

Input: A set D of unit-disks in $\mathbb{R}^2$.
Output: A partition $\{D_1, \ldots, D_\ell\}$ of D that minimizes

$$\sum_{i=1}^{\ell} |\text{ConvexHull}(D_i)|.$$

Other variant:

k-cluster fencing problem

Input: A set P of points in $\mathbb{R}^2$.
Output: A partition $\{P_1, \ldots, P_k\}$ of P into k clusters that minimizes

$$\sum_{i=1}^{k} |\text{ConvexHull}(P_i)|.$$

A.k.a. min perimeter sum problem.

History

k-cluster fencing	$f(k)n^{O(k)}$	Capoyleas, Rote, Woeginger'91
k-cluster fencing	$f(k)n^{O(k)}$	Arkin, Khuller, Mitchell'93
2-cluster fencing	n^3	Mitchell and Wynters'91
2-cluster fencing	$n \log^4 n$	Abrahamsen, de Berg, Buchin, Mehr, Mehrabi'17
Unit-disk fencing	$\exp(n \log n)$	Arkin, Khuller, Mitchell'93: Run the algorithm for k-cluster, for each $1 \leq k \leq n$.

Conjecture by Arkin, Khuller, Mitchell'93:

k-cluster and unit-disk fencing are NP-Hard.

Our Results

Algorithms:

Unit-disk fencing	$n \operatorname{poly} \log n$
k-cluster fencing	n^{27}

Unit-disk algorithm extends to polygons.

Take-home message

Unit-disk fencing: Nice structure, simple polynomial time algorithms, more complicated near-linear time algorithm.

k-cluster fencing: Nice separator properties, not NP-Hard!

This talk: A polynomial time algorithm for unit-disk fencing

1. Simplify the problem
2. Structural property
3. Algorithm

Our problem for the rest of the talk

Input: A set P of points in $\mathbb{R}^2$. An opening cost c.
Output: A partition $\{P_1, \ldots, P_\ell\}$ of P that minimizes

$$\ell \cdot c + \sum_{i=1}^{\ell} \text{ConvexHull}(P_i).$$

Claim: Solving this problem helps us solve the unit-disk fencing problem.

Formally: For any instance of the unit-disk fencing problem D, define an instance of the new problem:

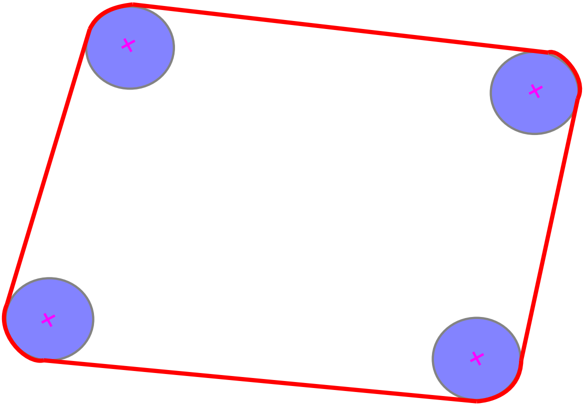
- c = perimeter of unit-disk
- $P = \{d \mid d$ center of a disk in $D\}$.

Any optimal solution for this problem is also an optimal solution for the unit-disk fencing problem.

- c = perimeter of unit-disk
- $P = \{d \mid d$ center of a disk in $D\}$.

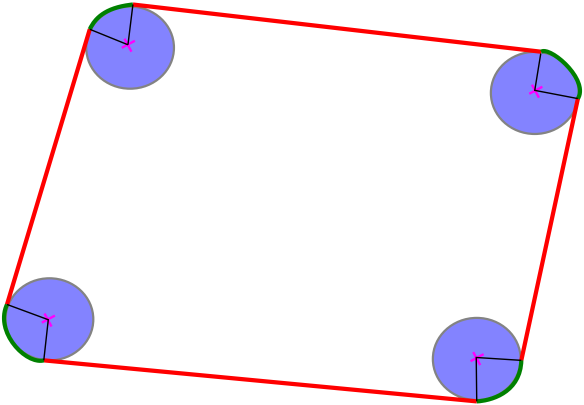
Any optimal solution for this problem is also an optimal solution for the unit-disk fencing problem.

Solution for unit-disk: Cost is length of red lines.



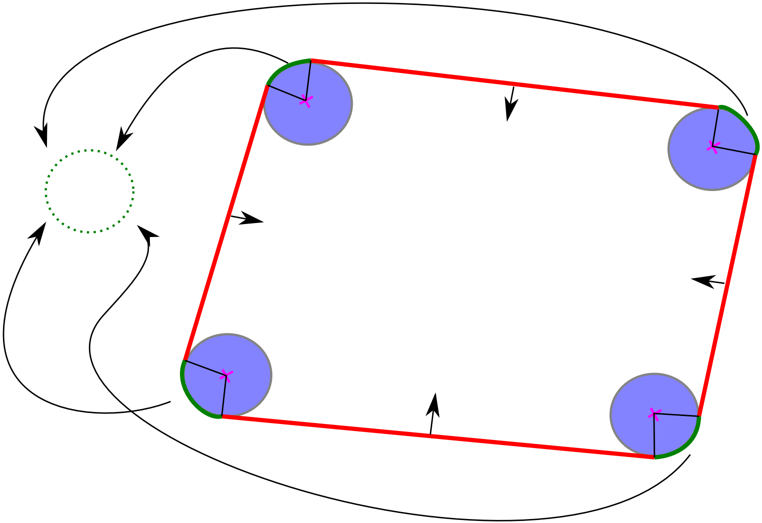
- c = perimeter of unit-disk
- $P = \{d \mid d$ center of a disk in $D\}$.

Any optimal solution for this problem is also an optimal solution for the unit-disk fencing problem.



- c = perimeter of unit-disk
- $P = \{d \mid d$ center of a disk in $D\}$.

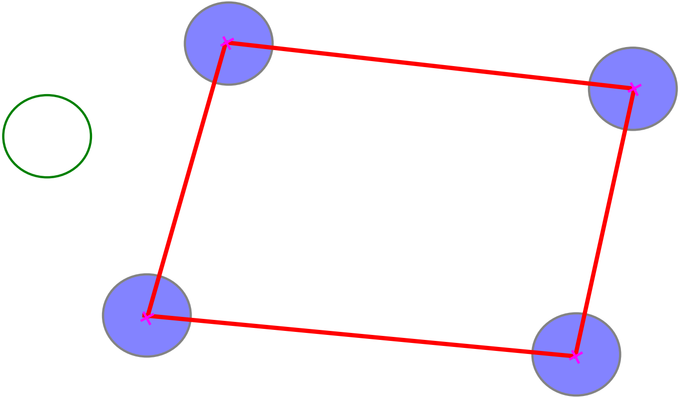
Any optimal solution for this problem is also an optimal solution for the unit-disk fencing problem.



- c = perimeter of unit-disk
- $P = \{d \mid d$ center of a disk in $D\}$.

Any optimal solution for this problem is also an optimal solution for the unit-disk fencing problem.

Solution for the new problem: cost is c + convexhull of centers of disks



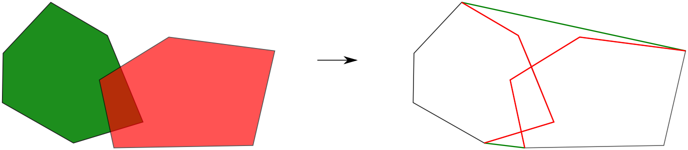
Our problem for the rest of the talk

Input: A set P of points in $\mathbb{R}^2$. An opening cost c.
Output: A partition $\{P_1, \ldots, P_\ell\}$ of P that minimizes

$$\ell \cdot c + \sum_{i=1}^{\ell} \text{ConvexHull}(P_i).$$

Obvious observation:

In an optimal solution, the convex hulls of the clusters do not intersect.



Key Definition:

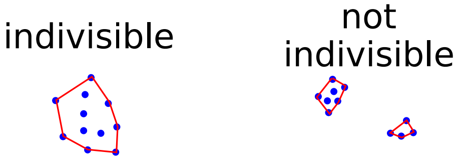
A set of points P is indivisible if the optimal solution for P is $\{P\}$.

Observation:

The clusters of OPT are indivisible.

Key Definition:

A set of points P is <u>indivisible</u> if the optimal solution for P is $\{P\}$.



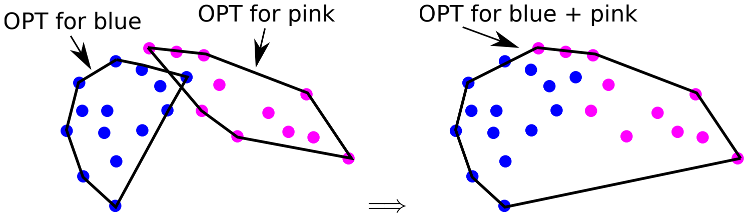
Observation:

The clusters of OPT are indivisible.

Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

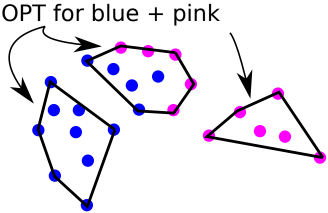


Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.

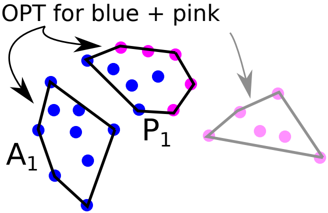


Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.



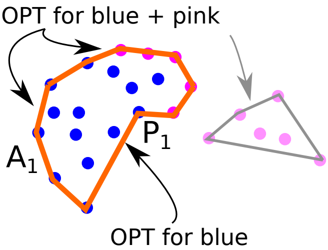
$A_1 \cup P_1$ is not indivisible.

Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.



$A_1 \cup P_1$ is not indivisible.

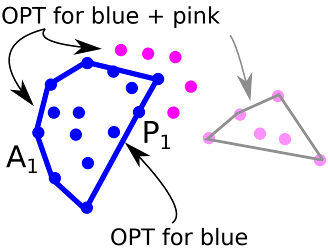
$$\text{Conv.Hull}(A_1 \cup P_1) \leq$$
$$\text{Conv.Hull(blue)} + \text{Conv.Hull}(P_1) - \text{Conv.Hull(red)}$$

Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.



$A_1 \cup P_1$ is not indivisible.

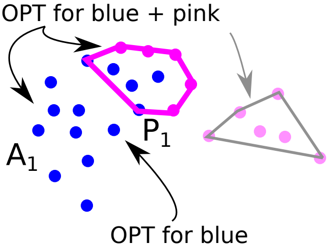
Conv.Hull$(A_1 \cup P_1) \leq$
Conv.Hull(blue) + Conv.Hull(P_1) − Conv.Hull(red)

Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.



$A_1 \cup P_1$ is not indivisible.

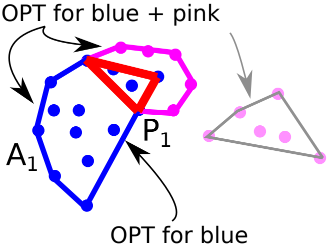
$$\text{Conv.Hull}(A_1 \cup P_1) \leq$$
$$\text{Conv.Hull(blue)} + \text{Conv.Hull}(P_1) - \text{Conv.Hull(red)}$$

Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.



$A_1 \cup P_1$ is not indivisible.

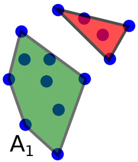
Conv.Hull$(A_1 \cup P_1) \leq$
Conv.Hull(blue) $+$ Conv.Hull(P_1) $-$ Conv.Hull(red)

Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.



$A_1 \cup P_1$ is not indivisible.

$\text{Conv.Hull}(A_1 \cup P_1) \leq$
$\text{Conv.Hull}(\text{blue}) + \text{Conv.Hull}(P_1) - \text{Conv.Hull}(\text{red})$

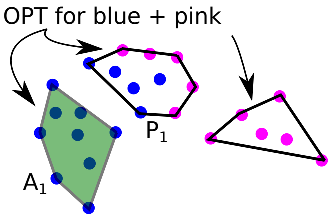
By indivisibility of blue:
$\text{OPT}(\text{blue}) = \text{Conv.Hull}(\text{blue}) + c \leq$
$\text{Conv.Hull}(A_1) + \text{Conv.Hull}(\text{red}) + 2c$

Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.



$A_1 \cup P_1$ is not indivisible.

$\text{Conv.Hull}(A_1 \cup P_1) \leq$
$\text{Conv.Hull(blue)} + \text{Conv.Hull}(P_1) - \text{Conv.Hull(red)}$

By indivisibility of blue:
$\text{OPT(blue)} = \text{Conv.Hull(blue)} + c \leq$
$\text{Conv.Hull}(A_1) + \text{Conv.Hull(red)} + 2c$

Summing up the two equations
$\text{Conv.Hull}(A_1 \cup P_1) + c \leq \text{Conv.Hull}(A_1) + \text{Conv.Hull}(P_1) + 2c$
Thus $A_1 \cup P_1$ is indivisible, a contradiction.

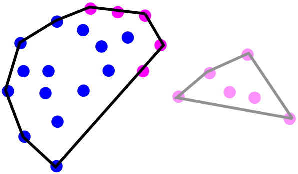
Proof of Structural Result

Lemma

Consider two indivisible sets of points A and B.
If their convex hulls intersect then $A \cup B$ is an indivisibile set of points.

Assume this is not true.

Solution better than
OPT for blue + pink



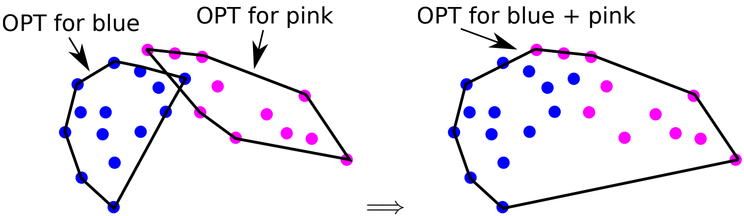
$A_1 \cup P_1$ is not indivisible.

$\text{Conv.Hull}(A_1 \cup P_1) \leq$
$\text{Conv.Hull}(\text{blue}) + \text{Conv.Hull}(P_1) - \text{Conv.Hull}(\text{red})$

By indivisibility of blue:
$\text{OPT}(\text{blue}) = \text{Conv.Hull}(\text{blue}) + c \leq$
$\text{Conv.Hull}(A_1) + \text{Conv.Hull}(\text{red}) + 2c$

Summing up the two equations
$\text{Conv.Hull}(A_1 \cup P_1) + c \leq \text{Conv.Hull}(A_1) + \text{Conv.Hull}(P_1) + 2c$
Thus $A_1 \cup P_1$ is indivisible, a contradiction.

This suggests the following algorithmic approach



Algorithm

Algorithm: Step 1

Recursively divide $\mathbb{R}^2$ into cells.

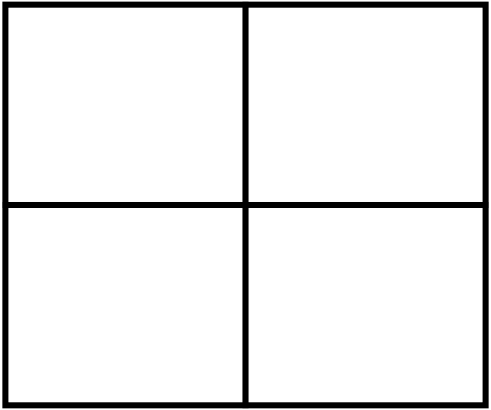
Level 0 cells.

Goal: Solve for cells of level i using solution for cells of level $i + 1$.

Algorithm: Step 1

Recursively divide $\mathbb{R}^2$ into cells.

Level 1 cells.

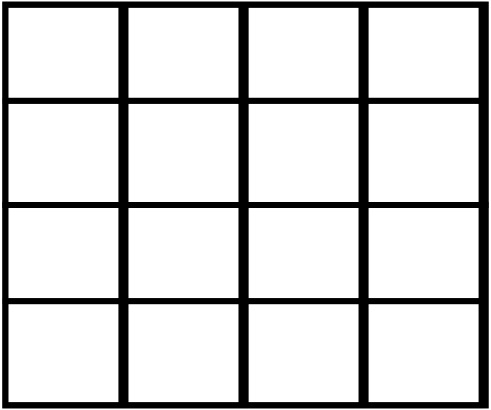


Goal: Solve for cells of level i using solution for cells of level $i + 1$.

Algorithm: Step 1

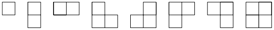
Recursively divide $\mathbb{R}^2$ into cells.

Level 2 cells.

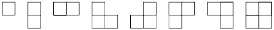


Goal: Solve for cells of level i using solution for cells of level $i + 1$.

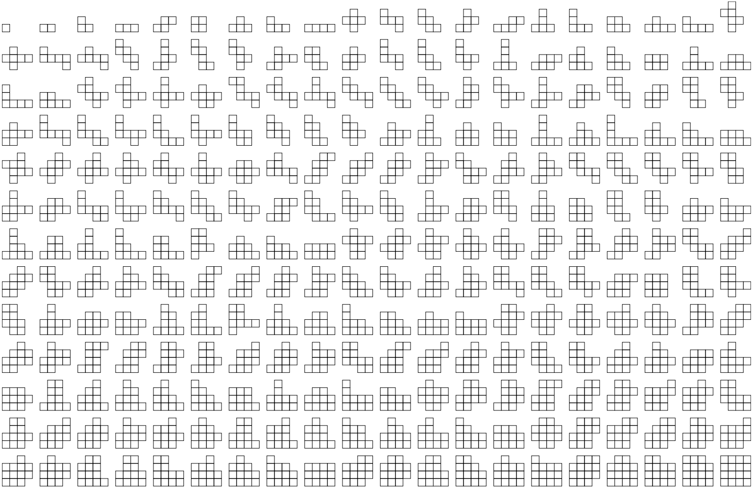
Define Shapes with Level-i Cells

Basic Polyominos: 

Define Shapes with Level-i Cells

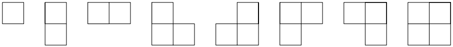
Basic Polyominos: 

Evolved Polyominos:



Property

Optimal solution for basic polyominoes made of level-i cells can be computed from optimal solutions for evolved polyominoes made of level-$i+1$ cells.



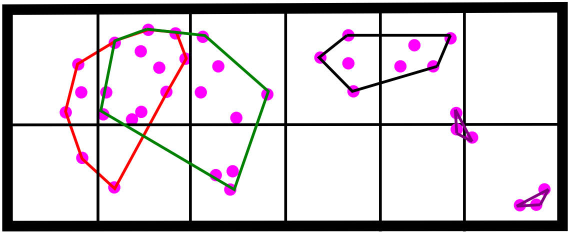
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



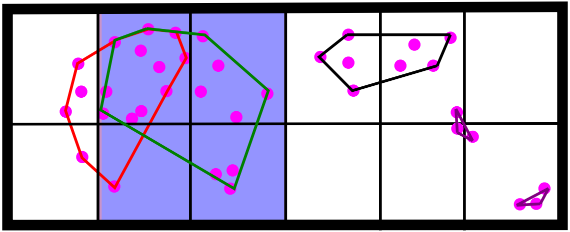
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



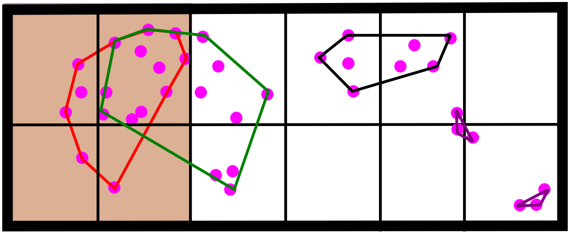
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



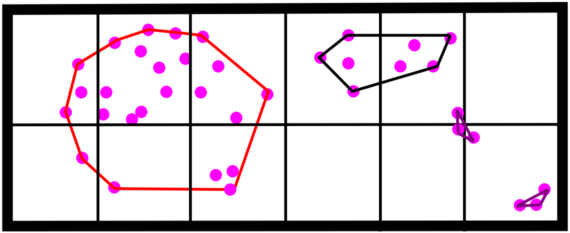
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



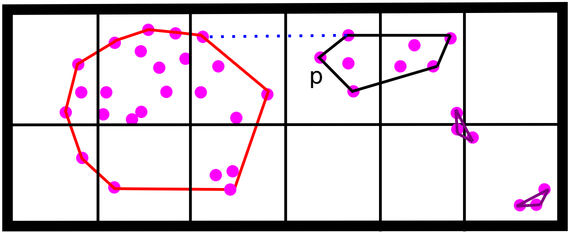
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



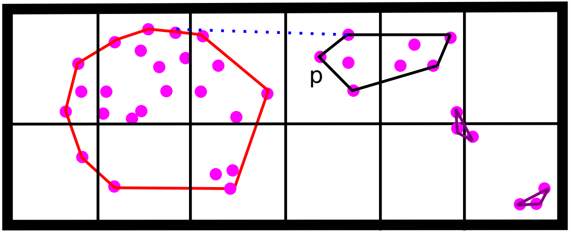
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



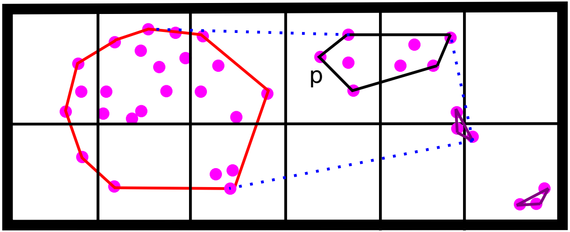
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



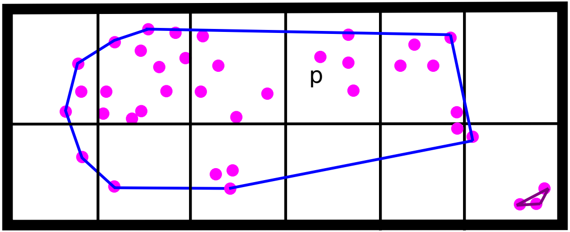
Property

Optimal solution for evolved polyominoes made of level-i cells computed:

- from optimal solution for basic polyominoes made of level-i cells, and
- Sweeping step to find out if there is a cluster intersecting all cells.

Alg:

1. Greedily merge optimal clusters of the basic polyominos that overlap.
2. Guess a point that belongs to the large cluster, compute the best cluster that intersects all cells, if it decreases the cost, keep it.



Overall Running Time

- $O(n^2)$ time per level
- $O(\log n)$ levels in total
- Overall running time : $O(n^2 \log n)$.

Faster

$O(n \text{ poly} \log n)$, using a more sophisticated algorithm for the last step.

k-Clustering Fencing Problem

Other structural result:

Separators with small complexity: Cells only intersect a few clusters of OPT.

Dynamic programming on cells.

Our Results

Algorithms:

Unit-disk fencing	$n \operatorname{poly} \log n$
k-cluster fencing	n^{27}

Unit-disk alg. extends to polygons w/ slightly worse complexity.

Take-home message

Unit-disk fencing: Nice structure, simple polynomial time algorithms, more complicated near-linear time algorithm.

k-cluster fencing: Nice separator properties, not NP-Hard!