Fixed-Parameter Algorithms for Covering Points with Lines

Apichat Heednacram
PhD Student

Supervisors: Prof. Vladimir Estivill-Castro
Dr. Francis Suraweera

Institute for Integrated & Intelligent Systems (IIIS),
Griffith University, Brisbane, Australia

Outline

- Research motivation
- Research aims
- The LINE COVER problem
- Other hard geometric problems and their progress

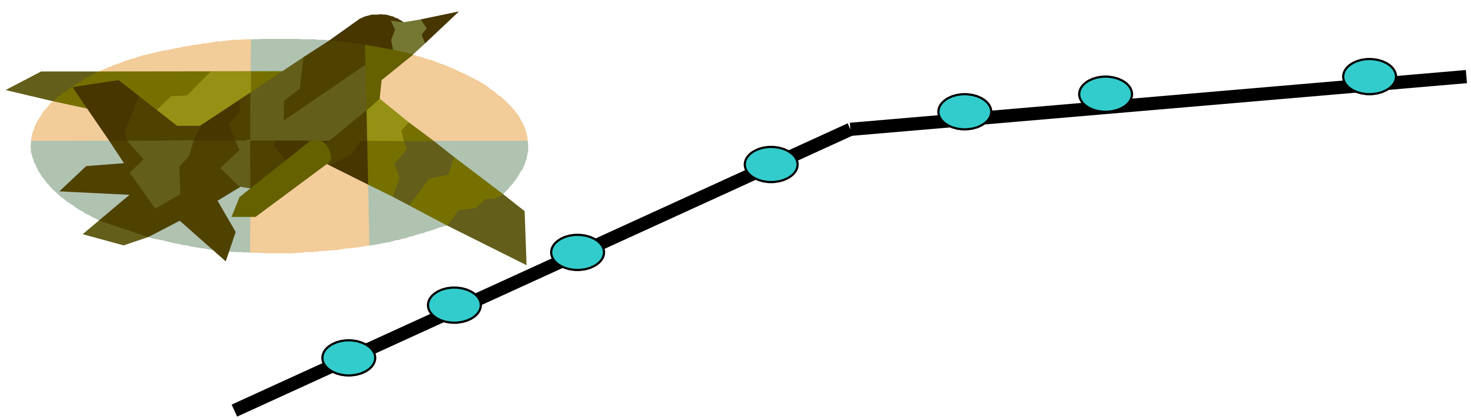
Research Motivation

- The survey of Giannopoulos et al. (2008) shows that there are only a few FPT results in computational geometry.
- Vankatesh Raman reports in “Parameterized Complexity Newsletter” (Sept, 2009) that there seems to be little research on parameterized techniques for geometric problems.

Research Aims

1. Solve some hard geometric problems (e.g. the LINE COVER problem, the Minimum Bends TSP, etc.).

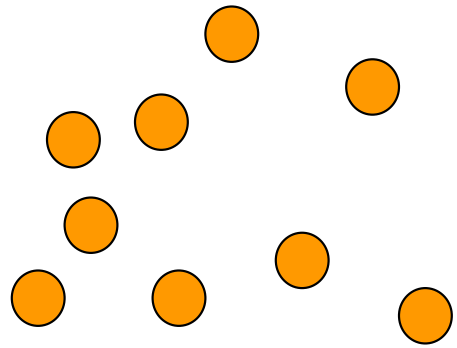
2. Design algorithms using FPT approach.



The LINE COVER Problem

Instance: A set *S* of *n* points, a positive integer *k*
Parameter: *k*
Question: Is it possible to cover *n* points in the plane with at most *k* lines?

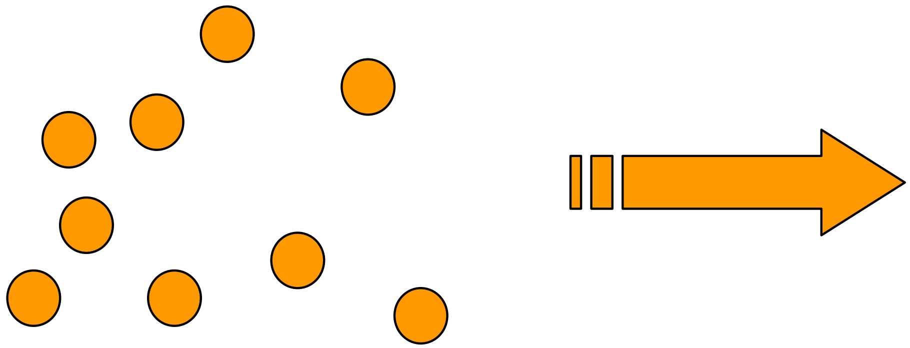
$n = 9, k = 3$



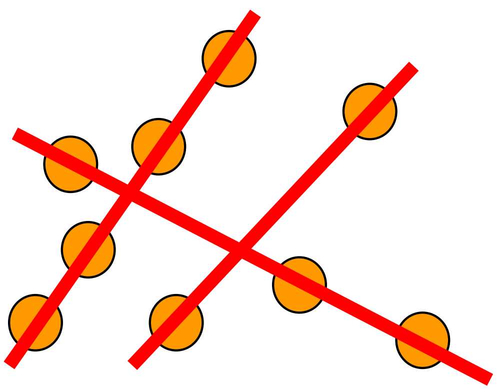
The LINE COVER Problem

Instance: A set *S* of *n* points, a positive integer *k*
Parameter: *k*
Question: Is it possible to cover *n* points in the plane with at most *k* lines?

$n = 9, k = 3$



YES, we can



History

- NP-hard (Megiddo and Tamir, 1982)
- FPT (Langerman and Morin, 2005)
- Improved time complexity (Grantson et al., 2006) based on
  - Guibas et al., 1996
  - Langerman and Morin, 2005

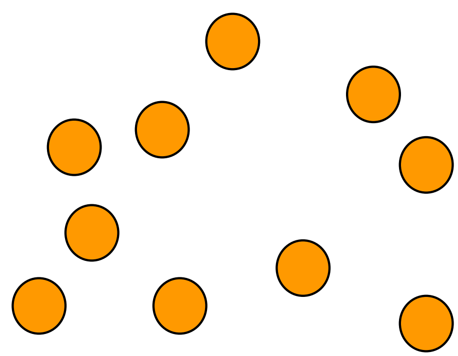
Our contributions

- New reduction rules
- Exact solutions for both the decision and optimization versions
- Experimental results and discussion in comparison with previous algorithms [LM05, GL06]
- Practical FPT algorithms

Reduction Rule 1

- If $k \geq \lceil n/2 \rceil$, then the answer is yes [GL06].

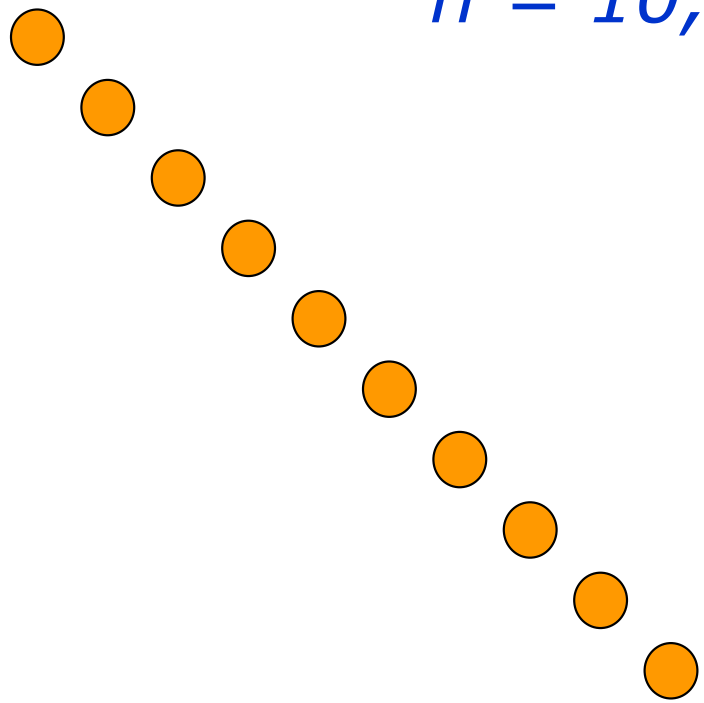
n = 10, k = 5



Reduction Rule 2

- If $k = 1$, then the answer is yes if and only if all points in S are co-linear [GL06].

$n = 10, k = 1$

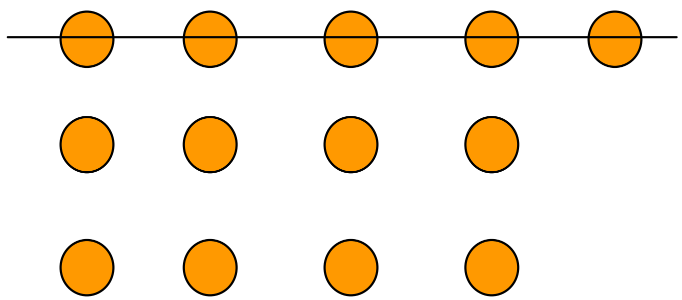


Reduction Rule 3

- If *S* has repeated points, then *S* can be simplified to a set with no repetitions and the answer for the simplified set is an answer for the original input [LM05].

Reduction Rule 4

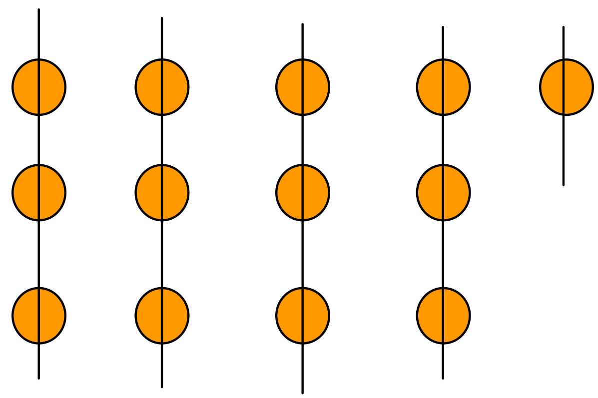
- If there is a set of k + 1 or more co-linear points, draw a line through them and put the points in the cover; remove these points from further consideration [LM05].



n = 13 k = 4

Reduction Rule 4

- If there is a set of k + 1 or more co-linear points, draw a line through them and put the points in the cover; remove these points from further consideration [LM05].

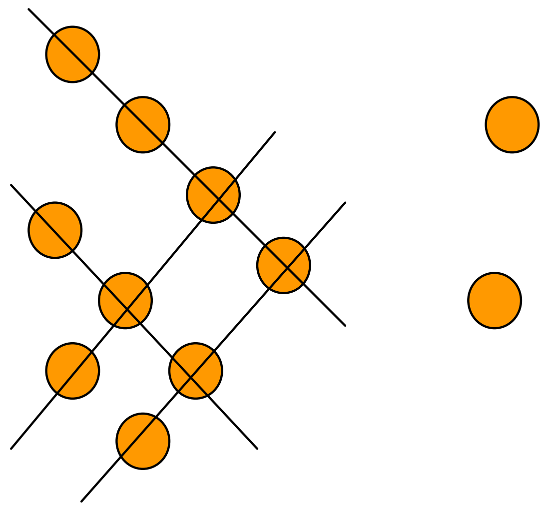


n = 13 k = 4

Can't cover with 4 lines

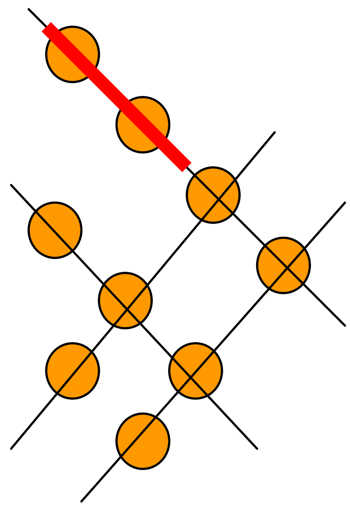
Reduction Rule 5

- Let L_3 be the set of all lines that cover at least 3 points in S and $cover(L_3)$ be all points in S cover by a line in L_3.
- Let $p_1 \neq p_2$ be two points in $S \backslash cover(L_3)$. Then, the original instance has a yes answer iff the instance $S \backslash \{p_1, p_2\}$ with parameter k-1 has a yes answer.



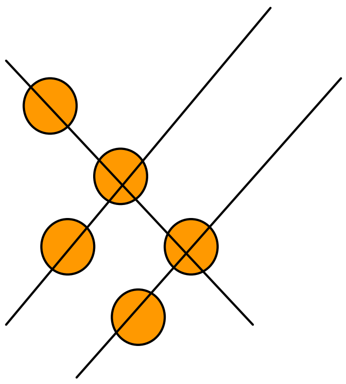
Reduction Rule 6

- Let $p_1 \neq p_2$ be two points in *cover*(L_3). Suppose that no other line in L_3 besides $\overline{p_1 p_2}$ covers p_1 or p_2. Then, the original instance has a yes answer iff the instance $S \setminus \{p_1, p_2\}$ with parameter k-1 has a yes answer.



Reduction Rule 6

- Let $p_1 \neq p_2$ be two points in $cover(L_3)$. Suppose that no other line in L_3 besides $\overline{p_1 p_2}$ covers p_1 or p_2. Then, the original instance has a yes answer iff the instance $S \setminus \{p_1, p_2\}$ with parameter k-1 has a yes answer.

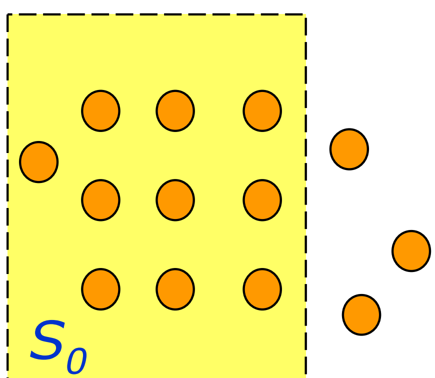


Kernelization

- What is kernelization?
- What is the problem kernel?
- The problem kernel of the LINE COVER problem has at most k^2 points. How?

Kernel Lemma

- If there is a subset S_0 of S so that $|S_0| \geq k^2+1$, the largest number of co-linear points is k, then the answer is no [GL06].
- **Proof:**
  - Each of these lines covered at most k points
  - Any cover of S_0 with k lines would cover at most k^2 points.
  - We cannot cover S_0, let alone S.



$n = 13 \quad k = 3$

$|S_0| = 10$

Example of Preprocessing Algorithm

Step 1: Application of Reduction Rule 1:
If $n \leq 2k$, answer YES and halt

Step 2: Application of Reduction Rule 2:
If all points in S are co-linear, answer YES and halt

Step 3: Application of Reduction Rule 4:
Collect at most k^2+1 points into a pool S_0 and remove a set of $k+1$ or more co-linear points.
If $|S_0| > k^2$, answer NO and halt

Step 4: Application of Reduction Rule 1:
If $n \leq 2k$, answer YES and halt

Step 5: Construct L_3, a set of all lines through 3 or more co-linear points.

Step 6: Application of Reduction Rule 5:
While there exist $\{p_1,p_2\} \in S \backslash cover(L_3)$,
do $S = S \backslash cover\overline{\{p_1,p_2\}}$, $k = k-1$

Step 7: Repeat all the above steps until every rule can no longer be applied

Example of Algorithm
PRIORITIZE POINTS IN HEAVY LINES

Step 1: (S',k') ← The Preprocessing Algorithm (S, k)

Step 2: Construct L_3, a set of all lines through 3 or more co-linear points in S'.

Step 3: while there exist $\{p_1,p_2\} \in cover(L_3)$, such that no other line in L_3 besides $\overline{p_1,p_2}$ covers p_1 or p_2 (Reduction Rule 6)

Step 4: do $S' = S' \backslash cover\{\overline{p_1,p_2}\}$, $k' = k'-1$

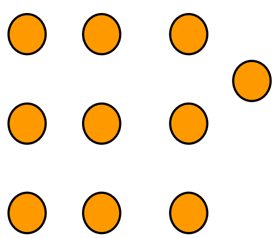
Step 5: return AROUNDTHEPOINTS(S',k')

Function AROUNDTHEPOINTS(*S'*,*k'*)

Reduction Rule:

If among all $\binom{n}{2}$ lines there is no line with at least $\lceil n/k \rceil$ points, then the answer is no.

$n = 10$



e.g.

$k = 2, \lceil n/k \rceil = 5$ Answer is NO

$k = 3, \lceil n/k \rceil = 4$ Answer is NO

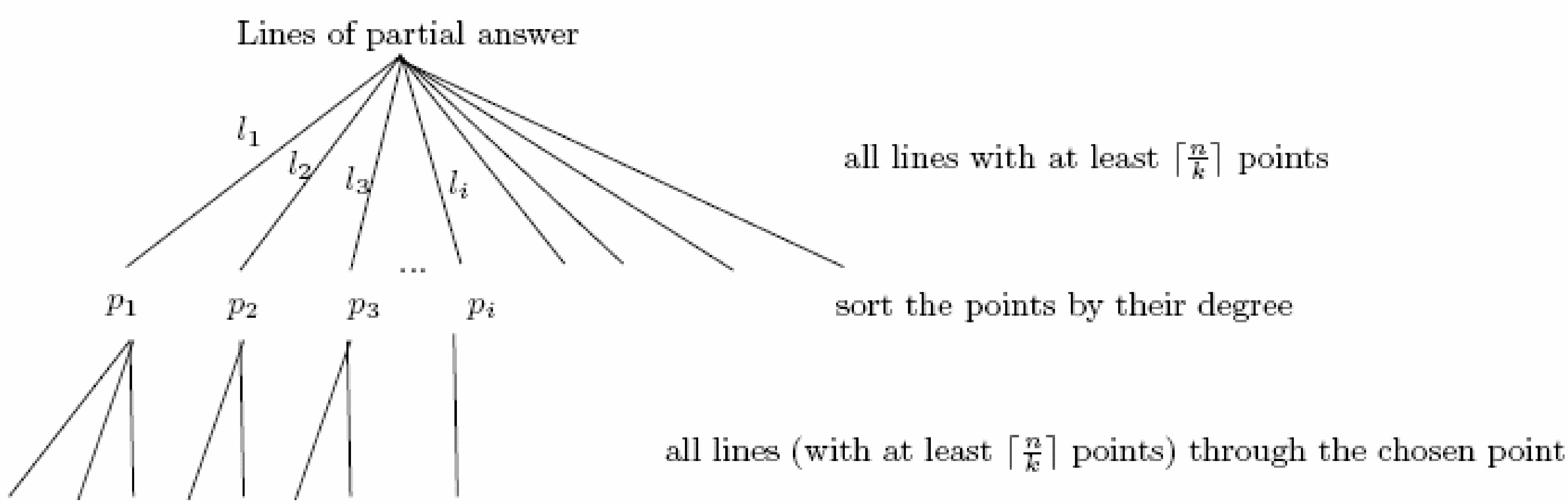
Function AROUNDTHEPOINTS(S',k')

- If S can be covered with k lines, then there is one of the k lines in the cover of S covering at least $\lceil n/k \rceil$ points.

- We make the following observation
  - There are $\leq 3k^2/2$ lines containing the average number of points.
  - There are $\leq 3k/2$ lines passing through a given point in the plane.

Function AROUNDTHEPOINTS(*S',k'*)

- We create a new bounded search tree



Algorithms for Decision Problem

Our algorithms:

1) CASCADE-AND-SORT	$O(n\log k + k^{2k+2})$
2) CASCADE-FURTHER	$O(n\log k + k^{2k+2})$
3) PRIORITIZE POINTS IN HEAVY LINES	$O(n\log k + k^4(k/2.22)^{2k})$

Previous algorithms [LM02]:

1) BST-DIM-SET-COVER	$O(k^{2k}n)$
2) KERNELIZE	$O(n^3 + k^{2k+2})$
3) CASCADE-AND-KERNELIZE	$O(n^3 + k^{2k+2})$

Algorithms for Decision Problem

Our algorithms:

1) CASCADE-AND-SORT $O(n\log k + k^{2k+2})$
2) CASCADE-FURTHER $O(n\log k + k^{2k+2})$
3) PRIORITIZE POINTS IN HEAVY LINES $O(n\log k + k^4(k/2.22)^{2k})$

CASCADE-AND-SORT sorts the points according to their weights before solving the problem kernel.

Algorithms for Decision Problem

Our algorithms:

1) CASCADE-AND-SORT $O(n\log k + k^{2k+2})$
2) CASCADE-FURTHER $O(n\log k + k^{2k+2})$
3) PRIORITIZE POINTS IN HEAVY LINES $O(n\log k + k^4(k/2.22)^{2k})$

CASCADE-AND-SORT sorts the points according to their weights before solving the problem kernel.

CASCADE-FURTHER includes one more rule (Reduction Rule 6) before solving the problem kernel.

Experimental results

- Table I: Average running times for the decision problem with 95% confidence intervals (10 random instances for each value of *k* and *n*).

Algorithm	k / n	Running Time		
	k	6		
	n	30	50	5000
Cascade-And-Sort		0.38 ± 0.03s	0.41 ± 0.05s	1.36 ± 0.07s
Cascade Further		0.40 ± 0.04s	0.41 ± 0.04s	1.33 ± 0.08s
Prioritize Points in Heavy Lines		0.41 ± 0.04s	0.44 ± 0.05s	1.34 ± 0.07s
BST-Dim-Set-Cover		223 ± 90s	525 ± 126s	≈ 15hr
Kernelize		2.71 ± 1.58s	0.17 ± 0.09s	6.81 ± 0.76s
Cascade-And-Kernelize		0.12 ± 0.06s	0.07 ± 0.03s	6.59 ± 0.61s

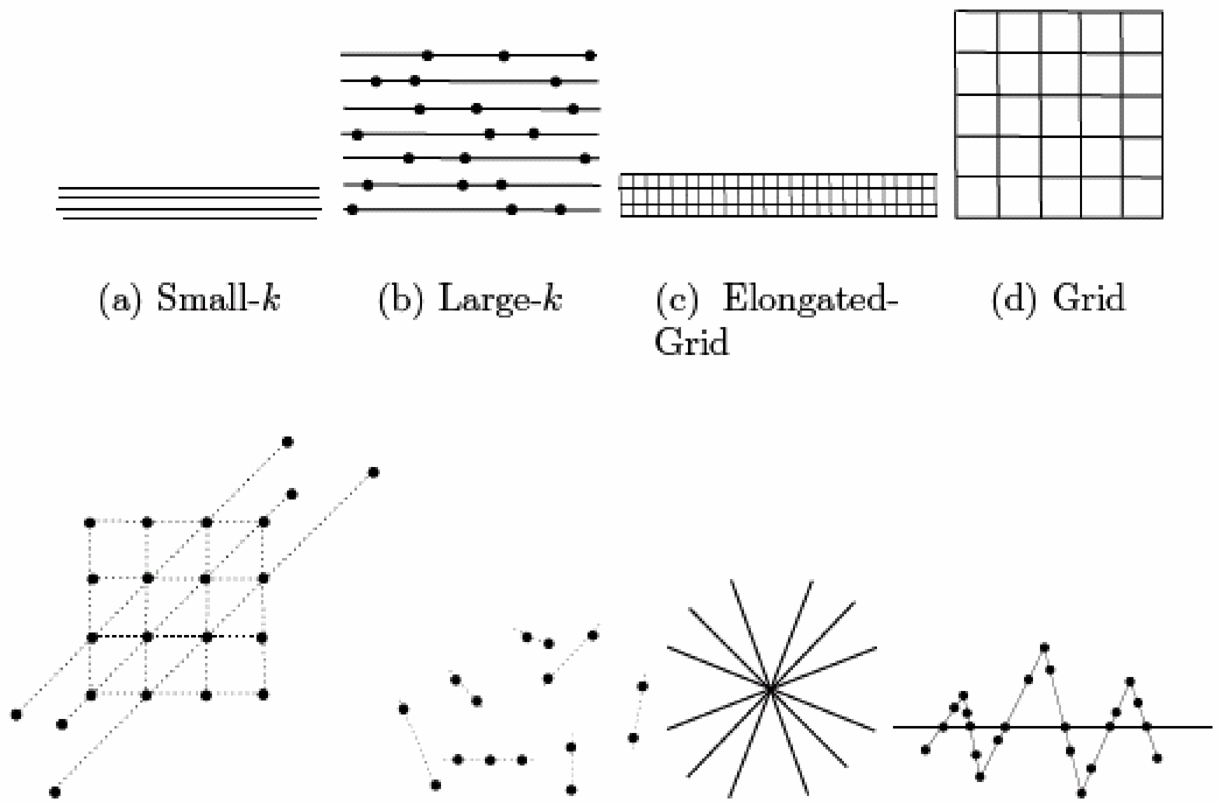
Experimental results

- Table II: Average running times for the decision problem with 95% confidence intervals (10 random instances for each value of *k* and *n*).

Algorithm	k / n	Running Time		
	k	7		
	n	30	50	5000
Cascade-And-Sort		0.98 ± 0.14s	0.70 ± 0.05s	1.71 ± 0.12s
Cascade Further		1.17 ± 0.19s	0.72 ± 0.05s	1.64 ± 0.10s
Prioritize Points in Heavy Lines		1.18 ± 0.20s	0.74 ± 0.07s	1.72 ± 0.13s
BST-Dim-Set-Cover		$\approx$ 3hr	$\approx$ 11hr	$\gg$ 24hr
Kernelize		$\approx$ 3hr	2.19 ± 1.99s	7.72 ± 0.51s
Cascade-And-Kernelize		$\approx$ 3hr	0.21 ± 0.11s	8.69 ± 1.05s

More results …

- We also evaluate when the structured instances are given



(a) Small-k (b) Large-k (c) Elongated-Grid (d) Grid

(e) Grid & Sparse (f) Sparse (g) Star (h) Wave

Algorithms for Optimization Problem

Our algorithms:

1) ONE INCREMENTS $O(nk\log k + k^4(k/2.22)^{2k})$
2) TAKE A GUESS $O(n^3 + k^4(k/2.22)^{2k})$
3) BINARY SEARCH $O(n\log^2 k + k^4\log k(k/1.11)^{4k})$

Previous algorithms [GL06]:

EXACTLINECOVER $O(n\log k + k^{2k+2})$

Algorithms for Optimization Problem

Our algorithms:

1) ONE INCREMENTS $O(nk\log k + k^4(k/2.22)^{2k})$
2) TAKE A GUESS $O(n^3 + k^4(k/2.22)^{2k})$
3) BINARY SEARCH $O(n\log^2 k + k^4\log k(k/1.11)^{4k})$

ONE INCREMENTS starts with $k' = 1$ and continue until we have $k' = k$.

Algorithms for Optimization Problem

Our algorithms:

1) ONE INCREMENTS $O(nk\log k + k^4(k/2.22)^{2k})$
2) TAKE A GUESS $O(n^3 + k^4(k/2.22)^{2k})$
3) BINARY SEARCH $O(n\log^2 k + k^4\log k(k/1.11)^{4k})$

ONE INCREMENTS starts with $k' = 1$ and continue until we have $k' = k$.

TAKE A GUESS improves the guess for the optimal k value based on reduction rules.

Algorithms for Optimization Problem

Our algorithms:

1) ONE INCREMENTS — $O(nk\log k + k^4(k/2.22)^{2k})$
2) TAKE A GUESS — $O(n^3 + k^4(k/2.22)^{2k})$
3) BINARY SEARCH — $O(n\log^2 k + k^4\log k(k/1.11)^{4k})$

ONE INCREMENTS starts with $k' = 1$ and continue until we have $k' = k$.

TAKE A GUESS improves the guess for the optimal k value based on reduction rules.

BINARY SEARCH doubles the k' value until we succeed before performing binary search for the optimal k value.

Algorithms for Optimization Problem

Our algorithms:

1) ONE INCREMENTS	$O(nk\log k + k^4(k/2.22)^{2k})$
2) TAKE A GUESS	$O(n^3 + k^4(k/2.22)^{2k})$
3) BINARY SEARCH	$O(n\log^2 k + k^4\log k(k/1.11)^{4k})$

Previous algorithms [GL06]:

EXACTLINECOVER	$O(n\log k + k^{2k+2})$

We omit the experimental results here.

Publication of the LINE COVER problem

- V. Estivill-Castro, A. Heednacram, and F. Suraweera. Reduction Rules Deliver Practical FPT-Algorithms for Covering Points and for TSP. *ACM Journal of Experimental Algorithmics*, 14:1.7--1.26, November 2009.

http://portal.acm.org/toc.cfm?id=1498698

Other hard geometric problems and their progress

1. The Rectilinear Minimum-Links Spanning Path in Higher Dimensions

   Instance: A set *S* of *n* points in R^d, a positive integer *k*
   Parameter: *k*
   Question: Is there a piecewise linear path through *n* points in *S* with at most *k* links (axis-parallel line-segments)?

   Our results: i) NP-complete, ii) FPT

2. The Rectilinear Hyperplane Cover in Higher Dimensions

   Instance: A set *S* of *n* points in R^d, a positive integer *k*
   Parameter: *k*
   Question: Is it possible to cover n points in S with at most *k* axis-parallel hyperplanes of d-1 dimensions?

   Our results: i) NP-complete, ii) FPT

Other hard geometric problems and their progress

3. The MBTSP in 2D

Instance: A set S of n points in 2D, a positive integer k
Parameter: k
Question: Is there a piecewise linear tour through n points in S with at most k bends?

Our result: FPT

4. The Rectilinear MBTSP in 2D

Instance: A set S of n points in 2D, a positive integer k
Parameter: k
Question: Is there a piecewise linear tour through n points in S with at most k bends where every line-segment in the path is either horizontal or vertical?

Our result: FPT

Conclusions

- The LINE COVER problem
  - new reduction rules
  - 3 algorithms for decision problem
  - 3 algorithms for optimization problem
  - Efficient FPT algorithms
- Other hard geometric problems
  - The Rectilinear Minimum-Links Spanning Path in Higher Dimensions
  - The Rectilinear Hyperplane Cover in Higher Dimensions
  - The MBTSP in 2D
  - The Rectilinear MBTSP in 2D

Thank you