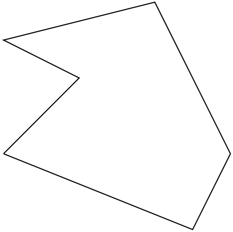
On the Travelling Salesperson Problem with Neighborhoods

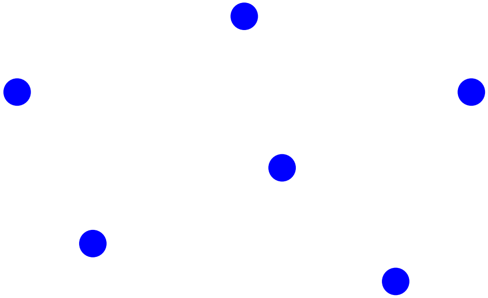
Antonios Antoniadis



June 28, 2019

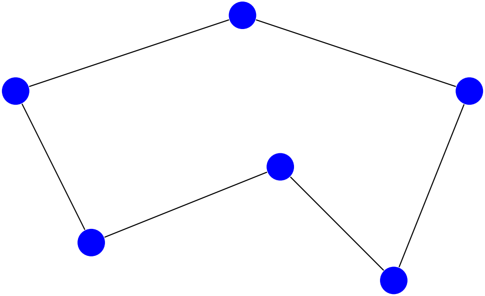
JPOC11

Travelling Salesperson Problem (TSP)



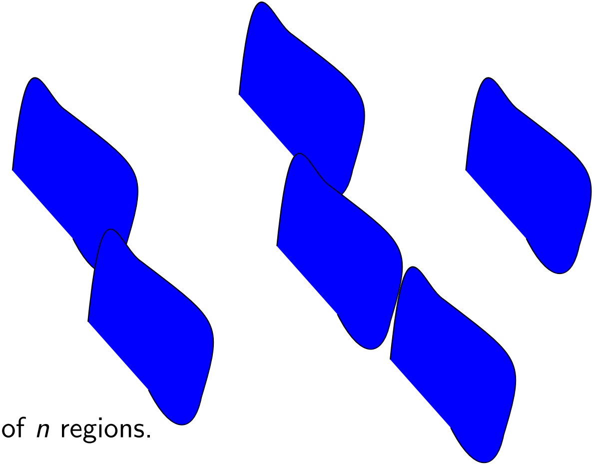
Input: A set of n points.

Travelling Salesperson Problem (TSP)



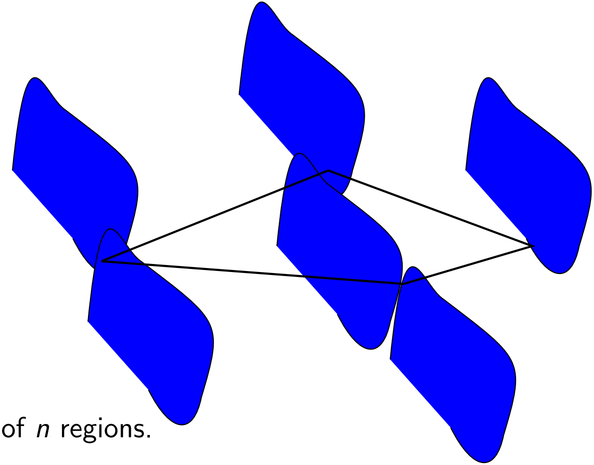
Input: A set of n points.
Output: A tour of minimum total length that visits all the points.

TSP with Neihgborhoods (TSPN)



Input: A set of n regions.

TSP with Neihgborhoods (TSPN)



Input: A set of n regions.

Output: A tour of minimum total length that visits all the regions.

Computational Complexity: TSP .vs. TSPN

	TSP	TSPN
exact solution	**NP-hard** Papadimitriou '77	**NP-hard** Papadimitriou '77
approximation	**Admits PTAS** Arora/Mitchell '96	**Does not admit PTAS** Gudmunsson & Levcopoulos '00

Definition

A polynomial-time algorithm ALG is called an α-approximation algorithm if cost(ALG,*I*)$\leq \alpha \cdot$cost(OPT,*I*) for all instances *I*

Definition

A PTAS is a family of algorithms $\{\text{ALG}_\epsilon\}_{\epsilon>0}$ such that for each $\epsilon > 0$, ALG_ϵ is a $(1+\epsilon)$-approximation algorithm.

Versions of TSPN

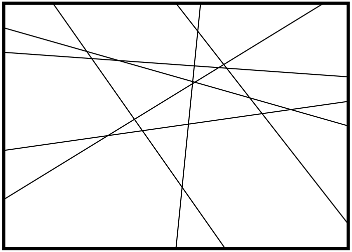
regions	**lower bound**	**upper bound**
k points (Group TSP) in $d = 2$	no const. apx. Safra & Schwarz '03	
$k = 2, d = 2$	no PTAS Dror & Orlin '03	
polygons in $d = 2$	no PTAS de Berg et al. '02	
conv. polytopes fixed d	NP-hard Papadimitriou '77	**Open Problem:** NP-hard? PTAS?
hyperplanes fixed d	**Open Problem:** NP-hard?	PTAS AA et al. '19
lines $d = 2$	-	$\in P$ Johnsson, '02
lines $d = 3$	NP-hard Papadimitriou '77	$\log^3 n$-apx. Dumitrescu& Tóth '13

Versions of TSPN

regions	**lower bound**	**upper bound**
k points (Group TSP) in $d = 2$	no const. apx. Safra & Schwarz '03	
$k = 2, d = 2$	no PTAS Dror & Orlin '03	
polygons in $d = 2$	no PTAS de Berg et al. '02	
conv. polytopes fixed d	NP-hard Papadimitriou '77	**Open Problem:** NP-hard? PTAS?
hyperplanes fixed d	**Open Problem:** NP-hard?	PTAS AA et al. '19
lines $d = 2$	-	$\in P$ Johnsson, '02
lines $d = 3$	NP-hard Papadimitriou '77	$\log^3 n$-apx. Dumitrescu& Tóth '13

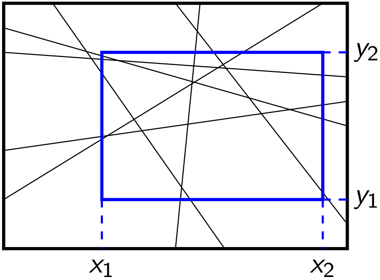
Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation



Linear Program:

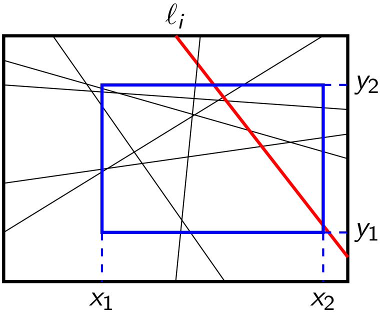
$$\min (x_2 - x_1) + (y_2 - y_1)$$

$$\text{s.t.} \quad x_1 \leq x_2$$

$$y_1 \leq y_2$$

Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation



Linear Program:

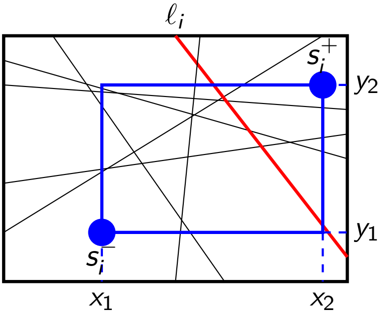
$$\min (x_2 - x_1) + (y_2 - y_1)$$

$$\text{s.t.} \quad x_1 \leq x_2$$

$$y_1 \leq y_2$$

Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation



Linear Program:

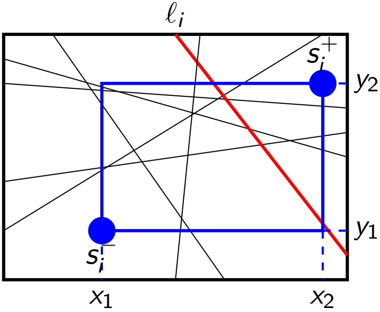
$$\min(x_2 - x_1) + (y_2 - y_1)$$

$$\text{s.t.} \quad x_1 \leq x_2$$

$$y_1 \leq y_2$$

Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

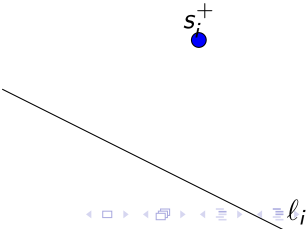


Linear Program:

$$\min (x_2 - x_1) + (y_2 - y_1)$$

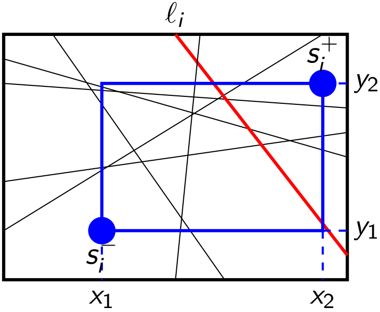
$$\text{s.t.} \quad x_1 \le x_2$$

$$y_1 \le y_2$$



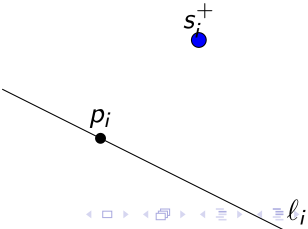
Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation



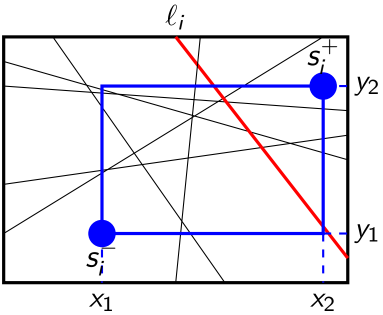
Linear Program:

$$
\begin{aligned}
& \min (x_2 - x_1) + (y_2 - y_1) \\
\text{s.t.} \quad & x_1 \le x_2 \\
& y_1 \le y_2
\end{aligned}
$$



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

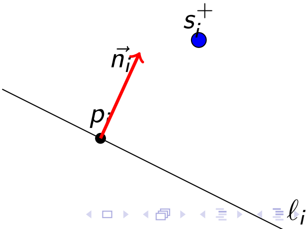


Linear Program:

$$\min(x_2 - x_1) + (y_2 - y_1)$$

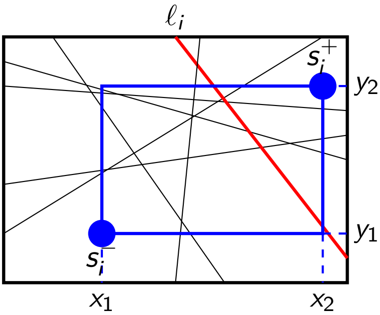
s.t. $x_1 \leq x_2$

$y_1 \leq y_2$



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

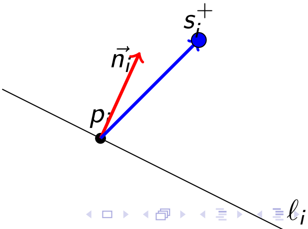


Linear Program:

$$\min(x_2 - x_1) + (y_2 - y_1)$$

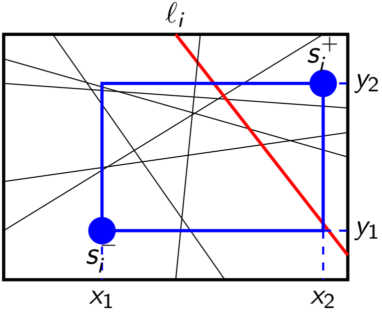
$$\text{s.t.} \quad x_1 \leq x_2$$

$$y_1 \leq y_2$$



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation



Linear Program:

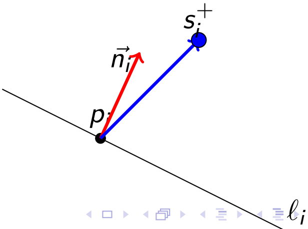
$$\min (x_2 - x_1) + (y_2 - y_1)$$

$$\text{s.t.} \quad x_1 \le x_2$$

$$y_1 \le y_2$$

$$\langle \vec{s_i^+} - \vec{p_i}, \vec{n_i} \rangle \ge 0$$

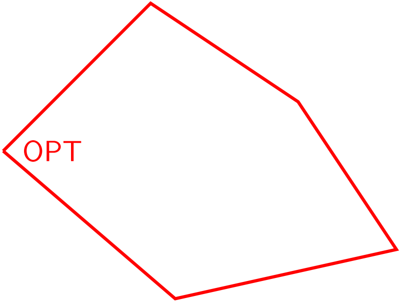
$$\langle \vec{s_i^-} - \vec{p_i}, \vec{n_i} \rangle \le 0$$



Hyperplane Neighborhoods, a Warmup

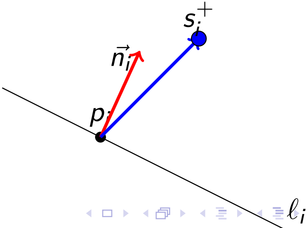
Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

Analysis



Linear Program:

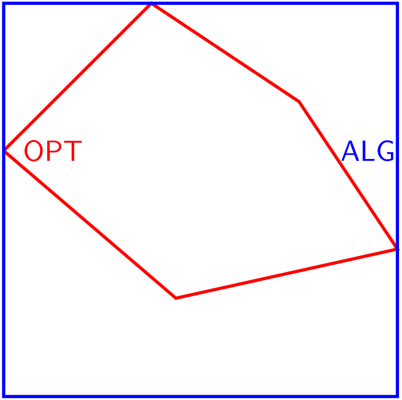
$$
\begin{aligned}
& \min (x_2 - x_1) + (y_2 - y_1) \\
\text{s.t.}\quad & x_1 \le x_2 \\
& y_1 \le y_2 \\
& \langle \vec{s_i^+} - \vec{p_i}, \vec{n_i} \rangle \ge 0 \\
& \langle \vec{s_i^-} - \vec{p_i}, \vec{n_i} \rangle \le 0
\end{aligned}
$$



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

Analysis



Linear Program:

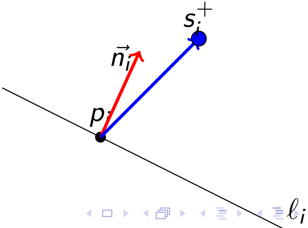
$$\min (x_2 - x_1) + (y_2 - y_1)$$

$$\text{s.t.} \quad x_1 \leq x_2$$

$$y_1 \leq y_2$$

$$\langle \vec{s_i^+} - \vec{p_i}, \vec{n_i} \rangle \geq 0$$

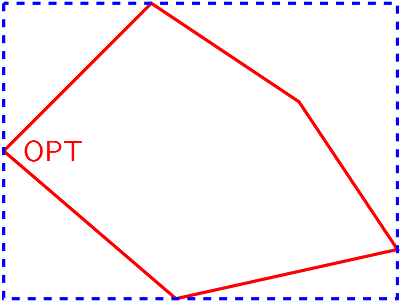
$$\langle \vec{s_i^-} - \vec{p_i}, \vec{n_i} \rangle \leq 0$$



Hyperplane Neighborhoods, a Warmup

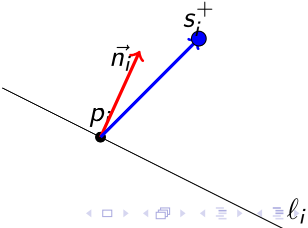
Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

Analysis



Linear Program:

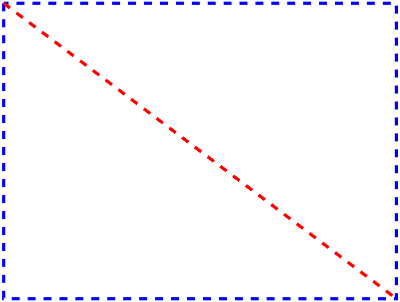
$$
\begin{aligned}
& \min (x_2 - x_1) + (y_2 - y_1) \\
\text{s.t.} \quad & x_1 \leq x_2 \\
& y_1 \leq y_2 \\
& \langle \vec{s_i^+} - \vec{p_i}, \vec{n_i} \rangle \geq 0 \\
& \langle \vec{s_i^-} - \vec{p_i}, \vec{n_i} \rangle \leq 0
\end{aligned}
$$



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

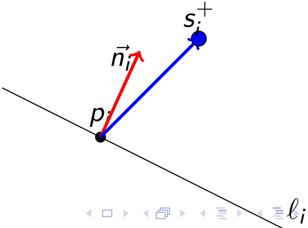
Analysis



Linear Program:

$$\min (x_2 - x_1) + (y_2 - y_1)$$

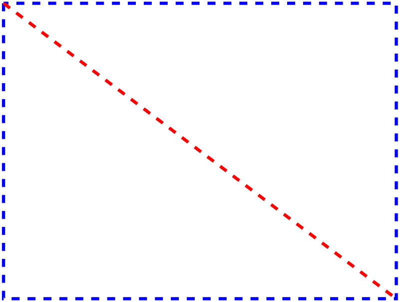
$$\begin{aligned} \text{s.t.} \quad & x_1 \le x_2 \\ & y_1 \le y_2 \\ & \langle \vec{s_i^+} - \vec{p_i}, \vec{n_i} \rangle \ge 0 \\ & \langle \vec{s_i^-} - \vec{p_i}, \vec{n_i} \rangle \le 0 \end{aligned}$$



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

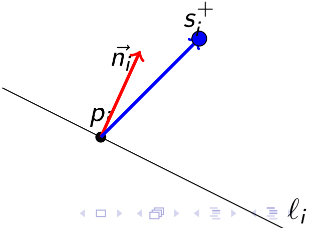
Analysis



$$\frac{\text{cost}(\text{ALG})}{\text{cost}(\text{OPT})} \leq \frac{\text{perim}(\text{box})}{\text{diag}(\text{box})} \in 2^{O(d)}$$

Linear Program:

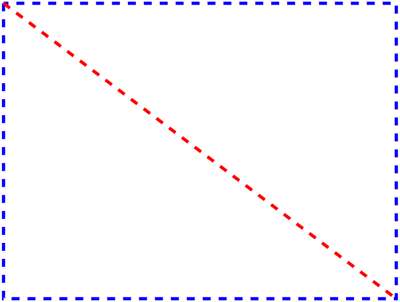
$$\begin{aligned} & \min (x_2 - x_1) + (y_2 - y_1) \\ \text{s.t.} \quad & x_1 \leq x_2 \\ & y_1 \leq y_2 \\ & \langle \vec{s_i^+} - \vec{p_i}, \vec{n_i} \rangle \geq 0 \\ & \langle \vec{s_i^-} - \vec{p_i}, \vec{n_i} \rangle \leq 0 \end{aligned}$$



Hyperplane Neighborhoods, a Warmup

Dumitrescu and Tóth, SODA '13: a $2^{\Theta(d)}$-approximation

Analysis

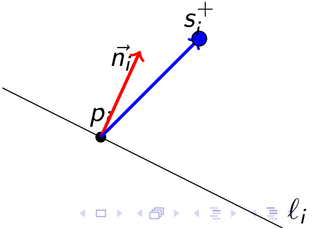


$$\frac{\text{cost(ALG)}}{\text{cost(OPT)}} \leq \frac{\text{perim(box)}}{\text{diag(box)}} \in 2^{O(d)}$$

Note: LP has constantly many variables $\to$ strongly polynomial linear time (Megiddo '84, Chan

Linear Program:

$$\begin{aligned} & \min (x_2 - x_1) + (y_2 - y_1) \\ \text{s.t.} \quad & x_1 \leq x_2 \\ & y_1 \leq y_2 \\ & \langle \vec{s_i^+} - \vec{p_i}, \vec{n_i} \rangle \geq 0 \\ & \langle \vec{s_i^-} - \vec{p_i}, \vec{n_i} \rangle \leq 0 \end{aligned}$$

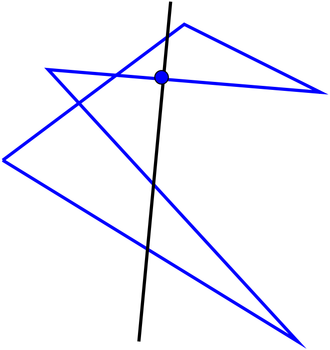


Idea for PTAS

Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.

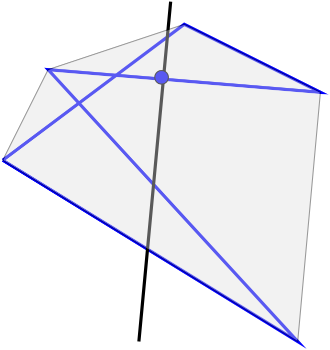
Idea for PTAS

Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.
$\Rightarrow$



Idea for PTAS

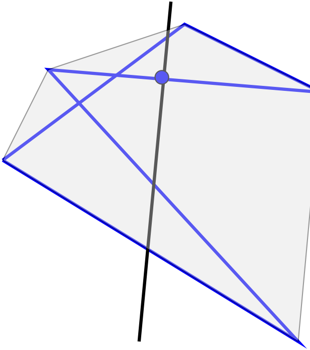
Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.
$\Rightarrow$



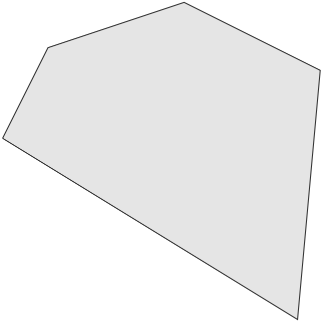
Idea for PTAS

Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.

$\Rightarrow$



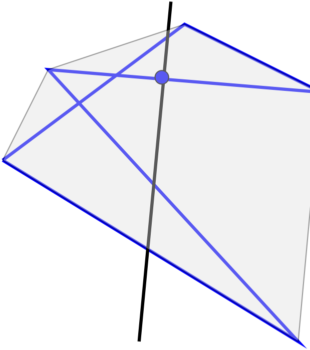
$\Leftarrow$



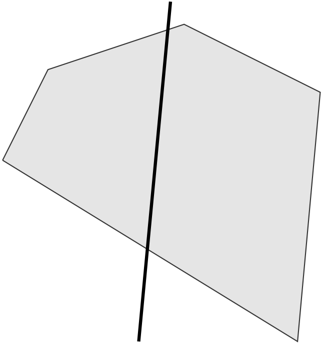
Idea for PTAS

Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.

$\Rightarrow$



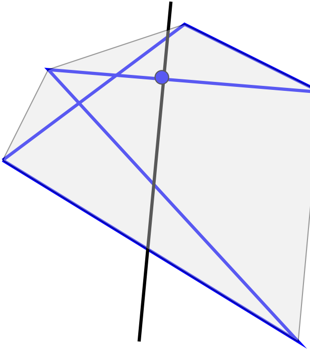
$\Leftarrow$



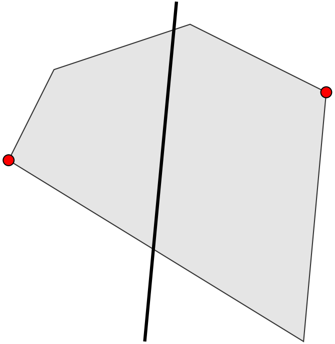
Idea for PTAS

Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.

$\Rightarrow$



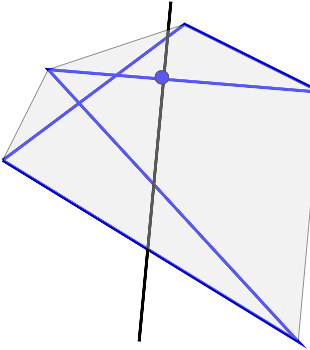
$\Leftarrow$



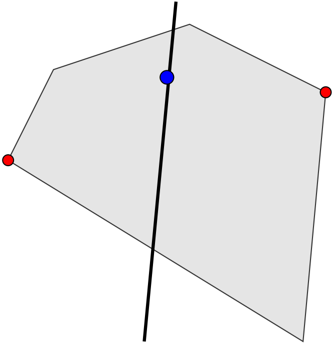
Idea for PTAS

Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.

$\Rightarrow$



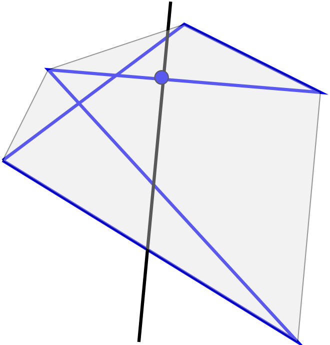
$\Leftarrow$



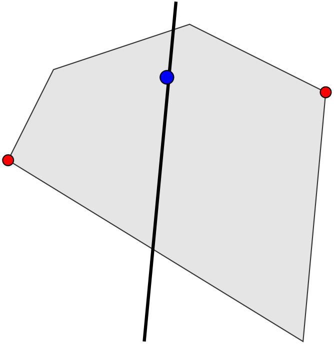
Idea for PTAS

Observation: Tour T is feasible $\Leftrightarrow$ conv(T) intersects all input hyperplanes.

$\Rightarrow$



$\Leftarrow$



Shortest tour visiting all vertices in conv(OPT) is again optimal!

Roadmap for PTAS

- Define polytopes whose complexity is bounded by a function of ϵ (and d).

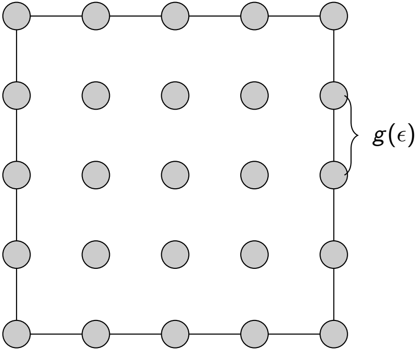
Roadmap for PTAS

- Define polytopes whose complexity is bounded by a function of ϵ (and d).
- Show that there is a $(1+\epsilon)$-approximation to OPT with a convex hull of bounded complexity.

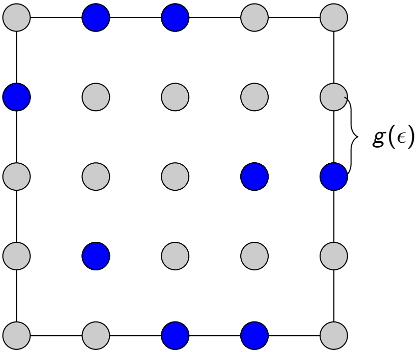
Roadmap for PTAS

- Define polytopes whose complexity is bounded by a function of ϵ (and d).
- Show that there is a $(1+\epsilon)$-approximation to OPT with a convex hull of bounded complexity.
- Use a linear program to find the "best" tour/polytope of bounded complexity.

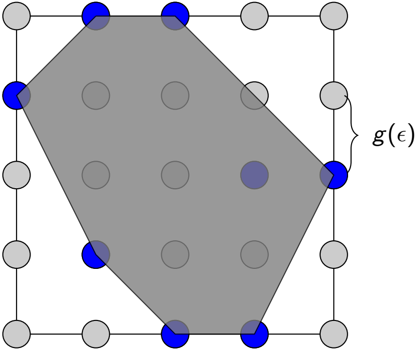
Bounded Complexity Polytopes



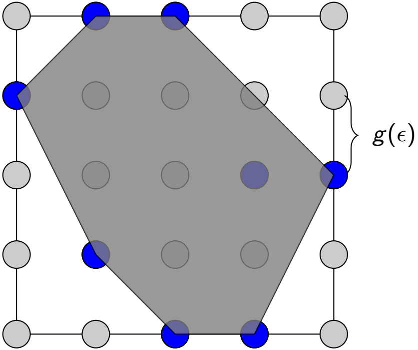
Bounded Complexity Polytopes



Bounded Complexity Polytopes



Bounded Complexity Polytopes



Scale & Translate

Bounded Complexity Polytopes suffice

Idea: Find a polytope P of bounded complexity so that $P \supseteq$ conv(OPT) and tsp(vertices(P))$\leq (1+\epsilon)\cdot$ cost(OPT).

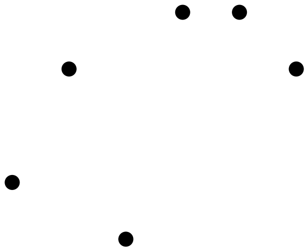
Two steps:

- **Sparcification** Find an intermediate polytope P' so that $P' \supseteq$ conv(OPT), tsp(vertices(P'))$\leq (1+\epsilon')\cdot$ cost(OPT), and P' has $O_{\epsilon,d}(1)$ many vertices.
- **Snapping:** Snap P' to the grid to obtain P, while increasing the tour length by at most another $(1+\epsilon'')$-factor.

Sparcification

Theorem (Chan '06)

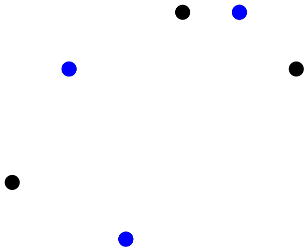
Given an m-point set in $\mathbb{R}^d$*, one can construct an* ϵ*-core-set of size* $O(1/\epsilon^{(d-1)/2})$ *for the extent measure.*



Sparcification

Theorem (Chan '06)

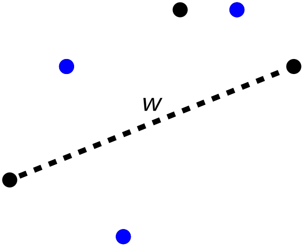
Given an m-point set in $\mathbb{R}^d$, one can construct an ϵ-core-set of size $O(1/\epsilon^{(d-1)/2})$ for the extent measure.



Sparcification

Theorem (Chan '06)

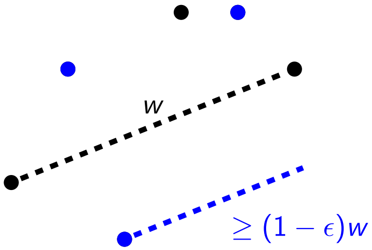
Given an m-point set in $\mathbb{R}^d$*, one can construct an* ϵ*-core-set of size* $O(1/\epsilon^{(d-1)/2})$ *for the extent measure.*



Sparcification

Theorem (Chan '06)

Given an m-point set in $\mathbb{R}^d$, one can construct an ϵ-core-set of size $O(1/\epsilon^{(d-1)/2})$ for the extent measure.



Sparcifying

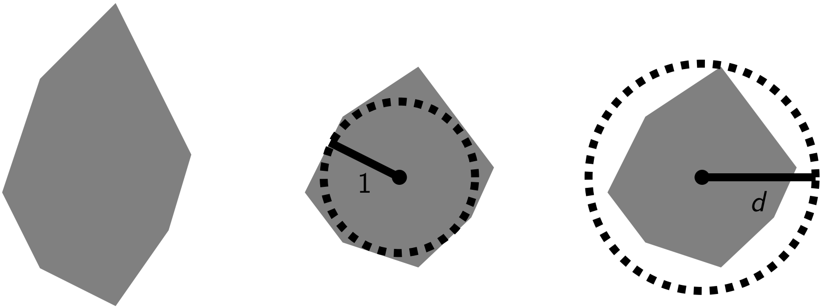
Theorem

Theorem (Ball '92) Let P be a convex polytope. If the volume-wise largest ellipsoid contained in P is $B(0,1)$, then $P \in B(0,d)$.

Sparcifying

Theorem

Theorem (Ball '92) Let P be a convex polytope. If the volume-wise largest ellipsoid contained in P is $B(0,1)$, then $P \in B(0,d)$.

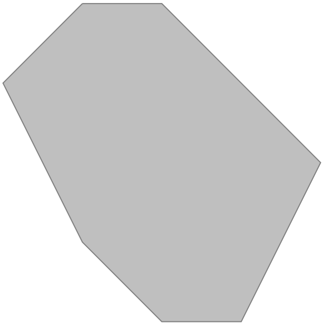


Sparcifying

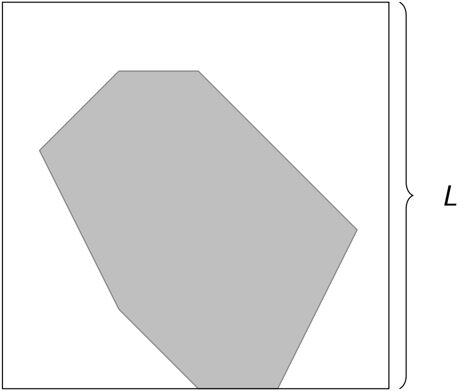
Theorem

Theorem We can select $O_{\epsilon,d}(1)$ *many vertices of OPT such that their convex hull scaled by a factor* $(1+\epsilon')$ *contains OPT.*

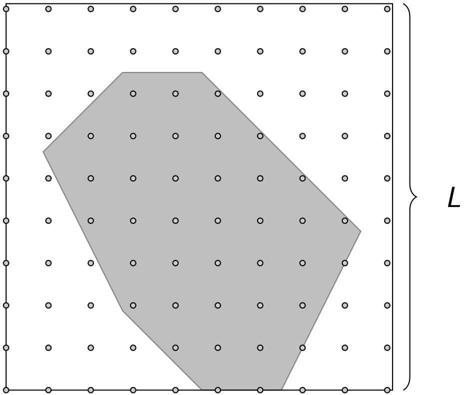
Snapping



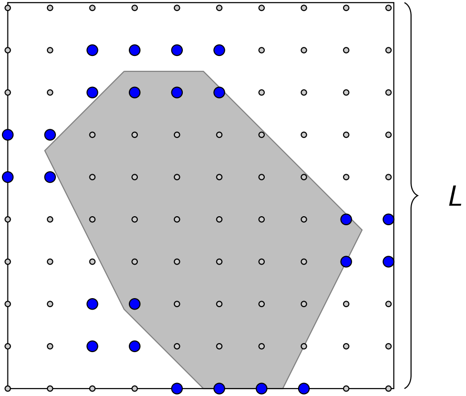
Snapping



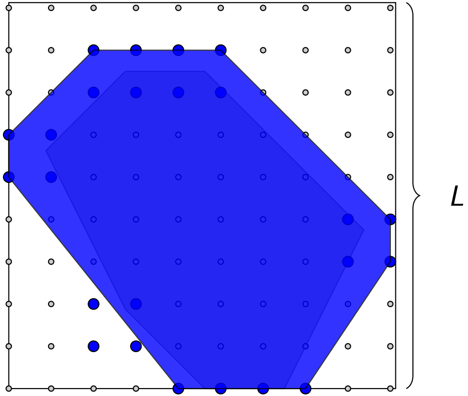
Snapping



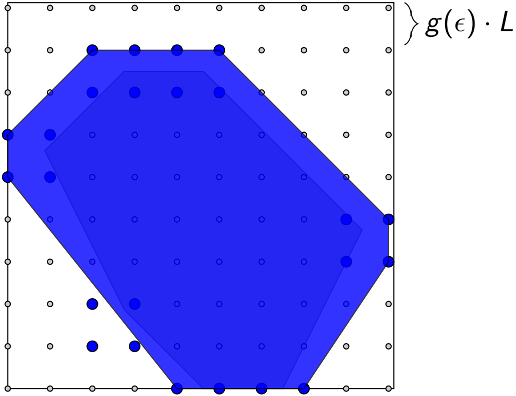
Snapping



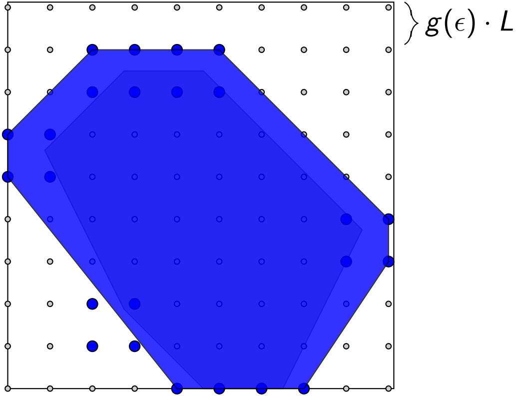
Snapping



Snapping

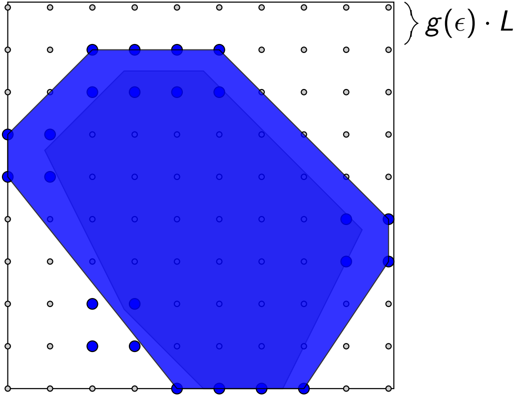


Snapping



Note: $L \leq \text{cost(OPT)}$

Snapping



Note: $L \leq \text{cost}(\text{OPT})$

We have:

- $P \supseteq P'$
- Detour cost: $O_\epsilon(1) \cdot 2^{O(d)} \cdot g(\epsilon) \cdot L \leq \epsilon \cdot \text{cost}(\text{OPT})$, for suitable $g(\epsilon)$.
- P is of bounded complexity.

Another Linear Program

"Guessing"

- The vertices of the polytope
- The order σ in which the tour (of length $\ell(\sigma)$) visits the vertices

Linear Program

- translation parameter $\vec{\rho}$
- scaling parameter λ

Another Linear Program

"Guessing"

- The vertices of the polytope
- The order σ in which the tour (of length $\ell(\sigma)$) visits the vertices

Note: For fixed vertices and σ, the tour length depends only on the scaling parameter λ.

Linear Program

- translation parameter $\vec{\rho}$
- scaling parameter λ

Another Linear Program

"Guessing"

- The vertices of the polytope
- The order σ in which the tour (of length $\ell(\sigma)$) visits the vertices

Note: For fixed vertices and σ, the tour length depends only on the scaling parameter λ.

Linear Program

- translation parameter $\vec{\rho}$
- scaling parameter λ

$$\min \lambda \cdot \ell(\sigma)$$

$$s.t.$$

$$\lambda > 1$$

$$\vec{\rho} \in \mathbb{R}^d$$

Another Linear Program

"Guessing"

- The vertices of the polytope
- The order σ in which the tour (of length $\ell(\sigma)$) visits the vertices

Note: For fixed vertices and σ, the tour length depends only on the scaling parameter λ.

Linear Program

- translation parameter $\vec{\rho}$
- scaling parameter λ

$$\min \lambda \cdot \ell(\sigma)$$

$$s.t.$$

$$\lambda > 1$$

$$\vec{\rho} \in \mathbb{R}^d$$

$$\langle \lambda s_i^+ - \rho, n_i \rangle \geq 0$$

$$\langle \lambda s_i^- - \rho, n_i \rangle \leq 0$$

Another Linear Program

"Guessing"

- The vertices of the polytope
- The order σ in which the tour (of length $\ell(\sigma)$) visits the vertices

Note: For fixed vertices and σ, the tour length depends only on the scaling parameter λ.

Linear Program

- translation parameter $\vec{\rho}$
- scaling parameter λ

$$\min \lambda \cdot \ell(\sigma)$$

s.t.

$$\lambda > 1$$

$$\vec{\rho} \in \mathbb{R}^d$$

$$\langle \lambda s_i^+ - \rho, n_i \rangle \geq 0$$

$$\langle \lambda s_i^- - \rho, n_i \rangle \leq 0$$

LP has (again) constantly many variables $\rightarrow$ strongly polynomial linear time (Megiddo '84, Chan '06)

Summing up...

Theorem

TSP with hyperplane neighborhoods admits a PTAS with strongly polynomial linear running time.

Summing up...

Theorem

TSP with hyperplane neighborhoods admits a PTAS with strongly polynomial linear running time.

...still the constant is exponential in d and doubly exponential in $1/\epsilon$.

The $d = 2$ Case

TSP with hyperplane neighborhoods in $d = 2$, a.k.a. TSP with lines in 2d, can be solved exactly in polynomial time through an elegant reduction to the watchman route problem.

The $d = 2$ Case

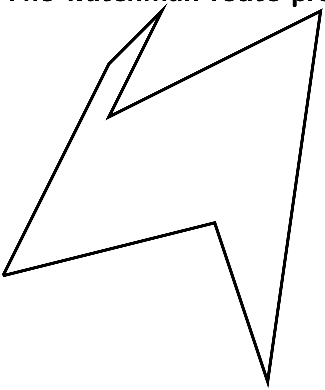
TSP with hyperplane neighborhoods in $d = 2$, a.k.a. TSP with lines in 2d, can be solved exactly in polynomial time through an elegant reduction to the watchman route problem.

The watchman route problem:

The $d = 2$ Case

TSP with hyperplane neighborhoods in $d = 2$, a.k.a. TSP with lines in $2d$, can be solved exactly in polynomial time through an elegant reduction to the watchman route problem.

The watchman route problem:

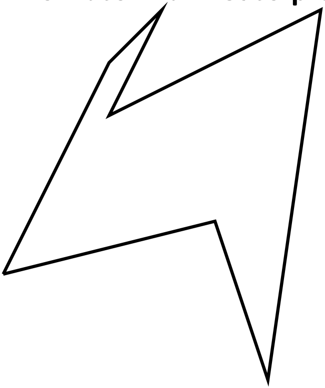


Given: A simple polygon P.

The $d = 2$ Case

TSP with hyperplane neighborhoods in $d = 2$, a.k.a. TSP with lines in 2d, can be solved exactly in polynomial time through an elegant reduction to the watchman route problem.

The watchman route problem:



Given: A simple polygon P.

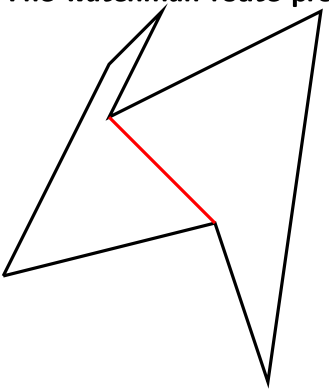
Output: A tour of minimal length, which

- remains in the interior of P, and
- "sees" every point $p \in P$.

The $d = 2$ Case

TSP with hyperplane neighborhoods in $d = 2$, a.k.a. TSP with lines in 2d, can be solved exactly in polynomial time through an elegant reduction to the watchman route problem.

The watchman route problem:

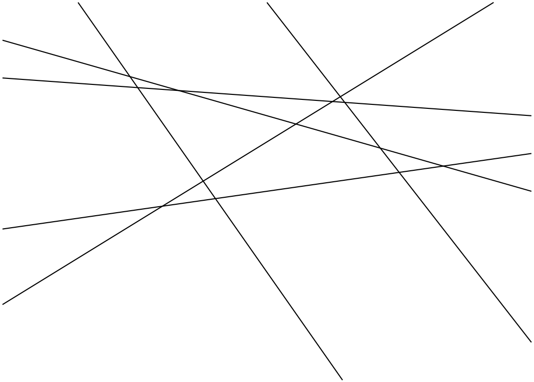


Given: A simple polygon P.

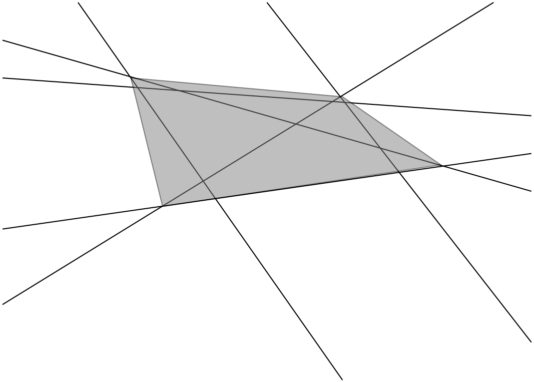
Output: A tour of minimal length, which

- remains in the interior of P, and
- "sees" every point $p \in P$.

A reduction to watchman route

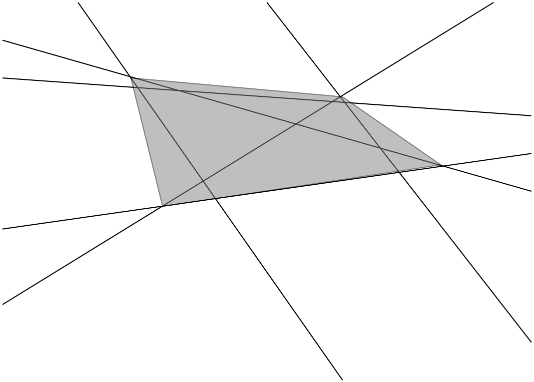


A reduction to watchman route



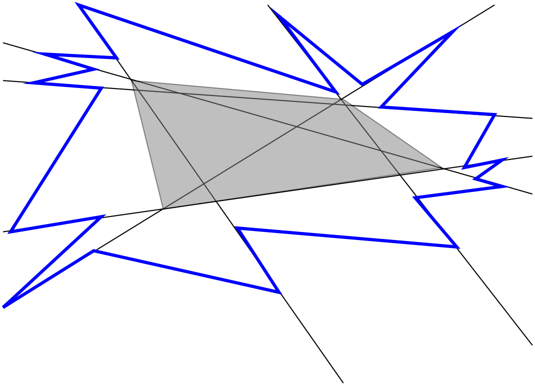
- Take convex hull of intersection points.

A reduction to watchman route



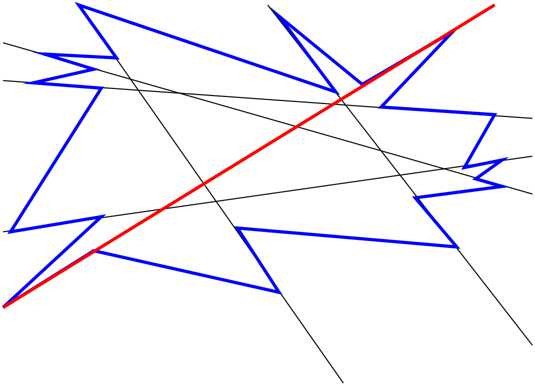
- Take convex hull of intersection points.
- Form "sawlike-structure".

A reduction to watchman route



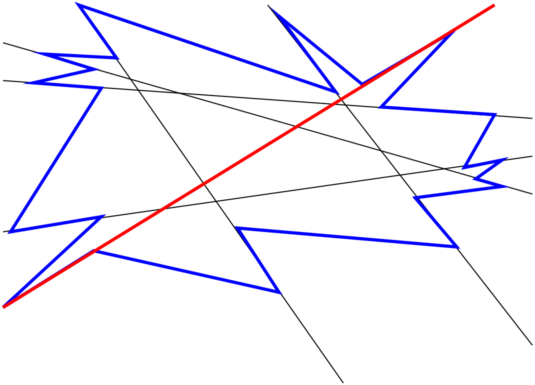
- Take convex hull of intersection points.
- Form "sawlike-structure".

A reduction to watchman route



- Take convex hull of intersection points.
- Form "sawlike-structure".

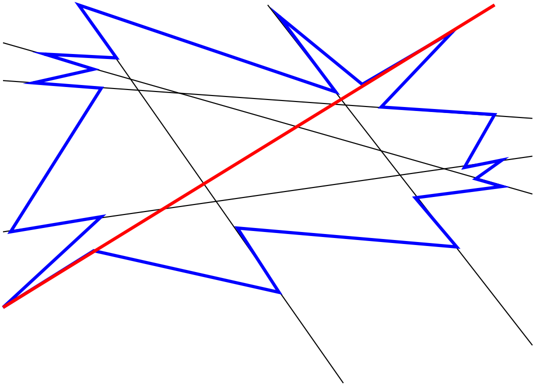
A reduction to watchman route



- Take convex hull of intersection points.
- Form "sawlike-structure".

A watchman route is feasible iff it "touches" all lines. (Johnsson '02)

A reduction to watchman route



- Take convex hull of intersection points.
- Form "sawlike-structure".

A watchman route is feasible iff it "touches" all lines. (Johnsson '02)

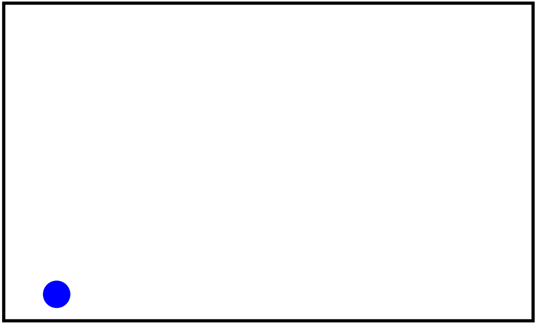
The watchman route problem is solvable in polynomial-time (Tan '01)

Online Setting

Hyperplane Chasing (Definition)

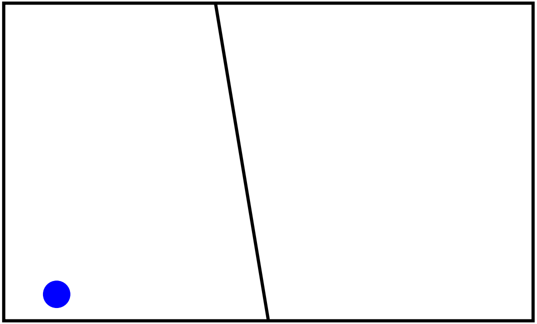
Input: An initial point p_0 and a sequence of hyperplanes
Output: A sequence of points, one per hyperplane
Cost: Total moved distance

Hyperplane Chasing (Definition)



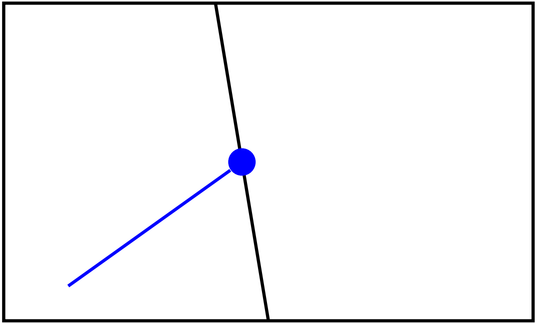
Input: An initial point p_0 and a sequence of hyperplanes
Output: A sequence of points, one per hyperplane
Cost: Total moved distance

Hyperplane Chasing (Definition)



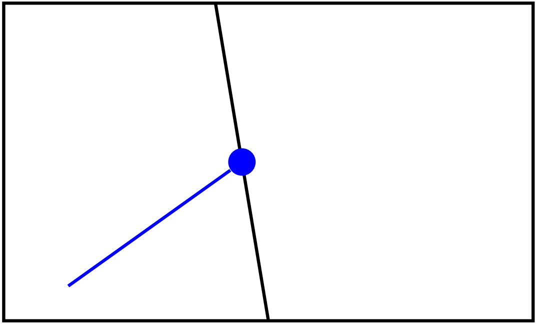
Input: An initial point p_0 and a sequence of hyperplanes
Output: A sequence of points, one per hyperplane
Cost: Total moved distance

Hyperplane Chasing (Definition)



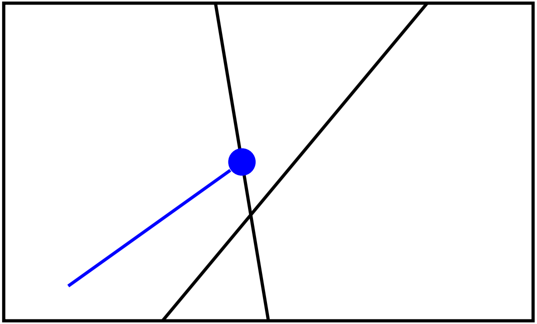
Input: An initial point p_0 and a sequence of hyperplanes
Output: A sequence of points, one per hyperplane
Cost: Total moved distance

Hyperplane Chasing (Definition)



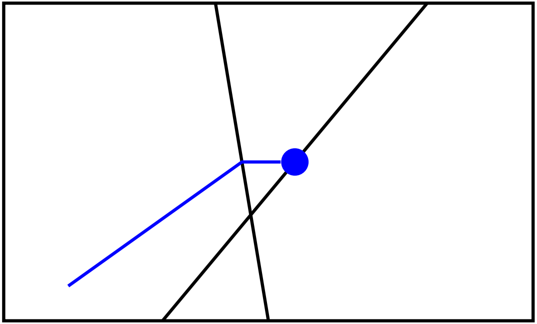
Input: An initial point p_0 and a sequence of hyperplanes
Output: A sequence of points, one per hyperplane
Cost: Total moved distance

Hyperplane Chasing (Definition)



Input: An initial point p_0 and a sequence of hyperplanes
Output: A sequence of points, one per hyperplane
Cost: Total moved distance

Hyperplane Chasing (Definition)



Input: An initial point p_0 and a sequence of hyperplanes
Output: A sequence of points, one per hyperplane
Cost: Total moved distance

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**

Move to intersection:
$\omega(1)$-**competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$**-competitive**

Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

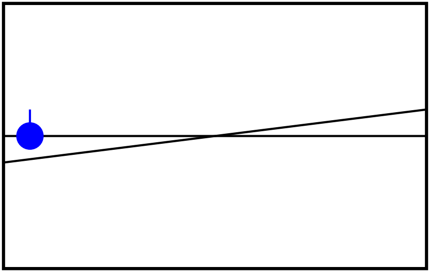
Greedy:
$\omega(1)$**-competitive**

Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$**-competitive**

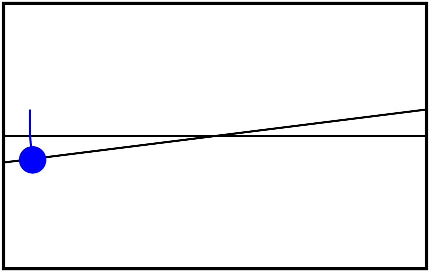


Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$**-competitive**

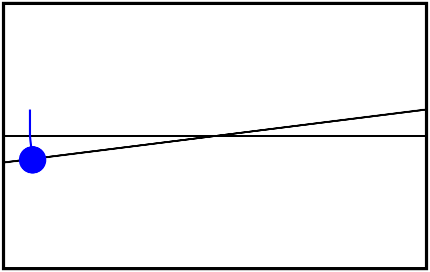


Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$**-competitive**

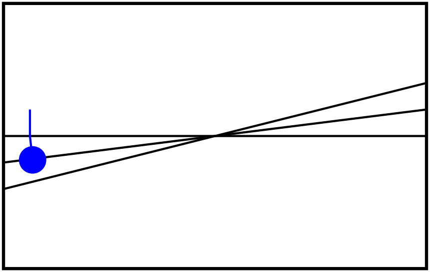


Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**

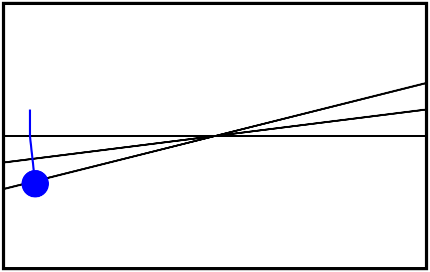


Move to intersection:
$\omega(1)$-**competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$**-competitive**

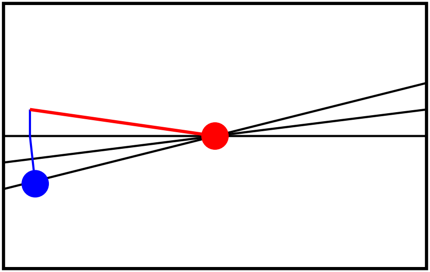


Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$**-competitive**

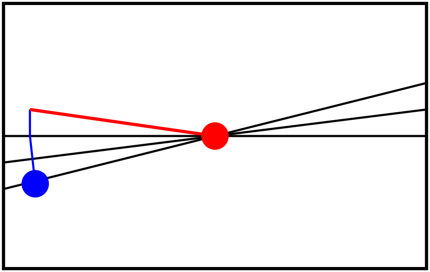


Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**

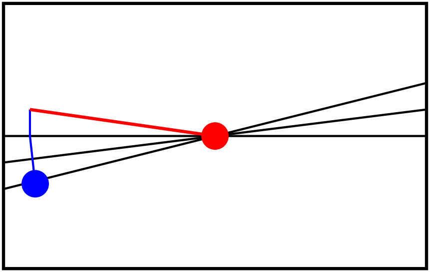


Move to intersection:
$\omega(1)$-**competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$**-competitive**

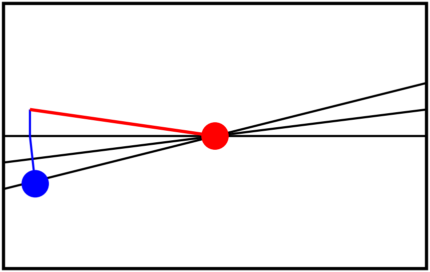


Move to intersection:
$\omega(1)$**-competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**

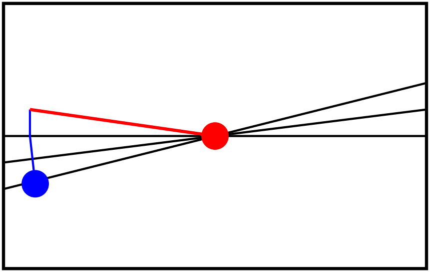


Move to intersection:
$\omega(1)$-**competitive**

Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**

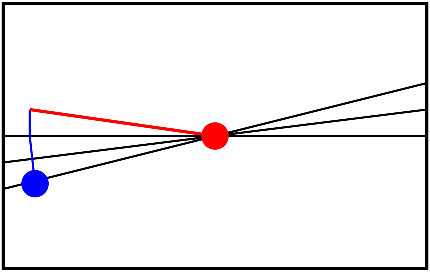


Move to intersection:
$\omega(1)$-**competitive**

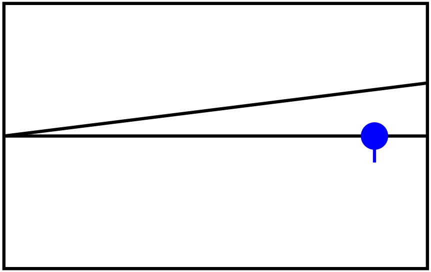
Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**



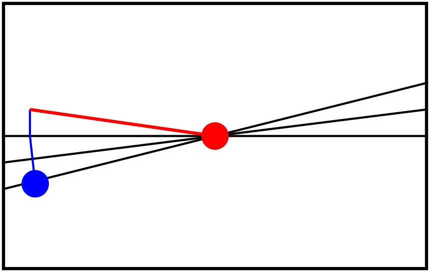
Move to intersection:
$\omega(1)$-**competitive**



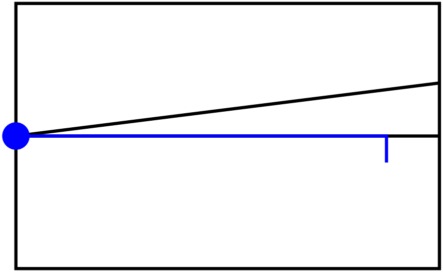
Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**



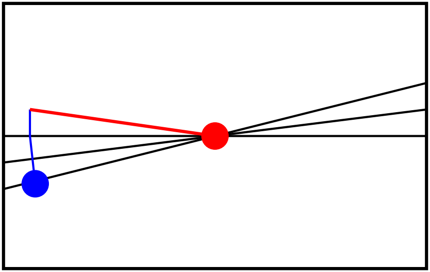
Move to intersection:
$\omega(1)$-**competitive**



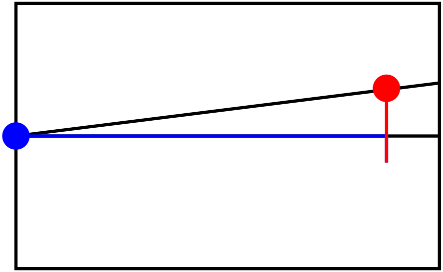
Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**



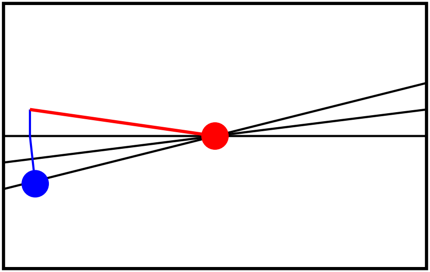
Move to intersection:
$\omega(1)$-**competitive**



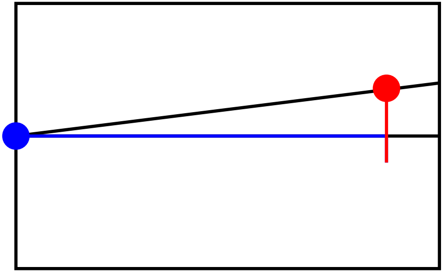
Line Chasing for $d = 2$

Simple Algorithms:

Greedy:
$\omega(1)$-**competitive**



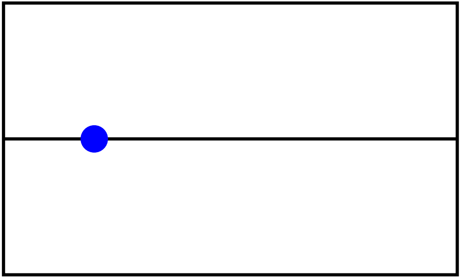
Move to intersection:
$\omega(1)$-**competitive**



Theorem (Friedman & Linial '93)

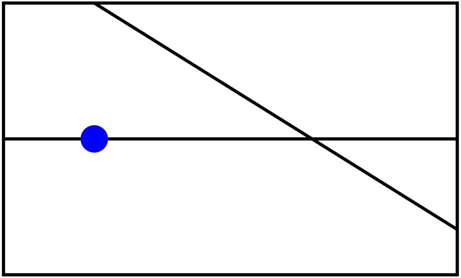
There is a $O(1)$-competitive algorithm for lines in $d = 2$.

Simpler $O(1)$-competitive algorithm (AA et al.'16)



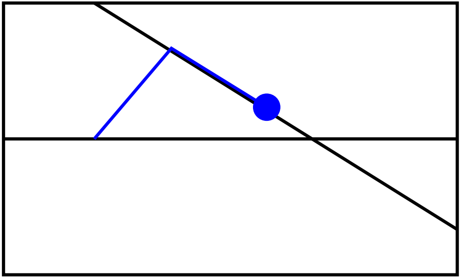
Algorithm:

Simpler $O(1)$-competitive algorithm (AA et al.'16)



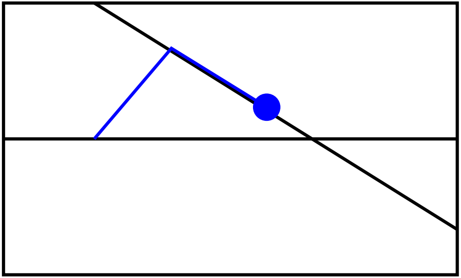
Algorithm:

Simpler $O(1)$-competitive algorithm (AA et al.'16)



Algorithm:
Move greedily to the next line, and then the same distance towards the intersection (if it exists).

Simpler $O(1)$-competitive algorithm (AA et al.'16)



Algorithm:
Move greedily to the next line, and then the same distance towards the intersection (if it exists).

Currently best algorithm: 3-competitive (Bienkowski et al. '18)

Convex Body Chasing

- Chasing convex bodies has connectionts to machine learning.

Convex Body Chasing

- Chasing convex bodies has connectionts to machine learning.
- Whether an $O(1)$-competitive algorithm for chasing convex bodies exists was a long-standing open problem.

Convex Body Chasing

- Chasing convex bodies has connectionts to machine learning.
- Whether an $O(1)$-competitive algorithm for chasing convex bodies exists was a long-standing open problem.
- Solved by Bubeck, Lee, Li, Sellke, STOC '19.

Convex Body Chasing

- Chasing convex bodies has connectionts to machine learning.
- Whether an $O(1)$-competitive algorithm for chasing convex bodies exists was a long-standing open problem.
- Solved by Bubeck, Lee, Li, Sellke, STOC '19.
- Simplified by Sellke & Argue, Gupta, Guruganesh and Tang, May '19

Future directions

What about:

- Is TSP with hyperplane neighborhoods NP-hard?
- Input hyperplanes have to be visited in a given order. Similar technique may work...
- Input is a set of lower dimensional affine subspaces. New techniques required...
- Best possible running time for $d = 2$?

Thanks!