k-Regret Minimizing Set

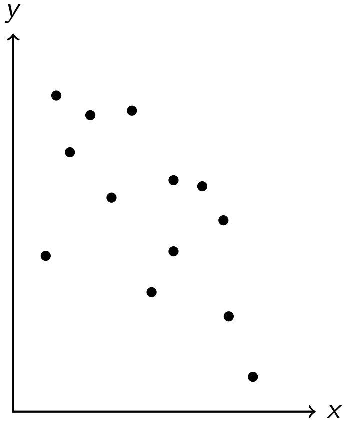
Efficient Algorithms and Hardness

Wei Cao[1] Jian Li[1] Haitao Wang[2] Kangning Wang[1]
Ruosong Wang[1] Raymond Chi-Wing Wong Wei Zhan[1]

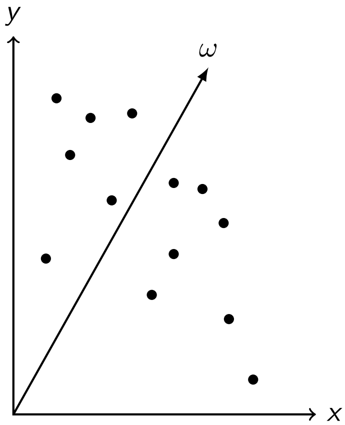
[1]Tsinghua University
[2]Utah State University
[3]The Hong Kong University of Science and Technology

ICDT, 2017

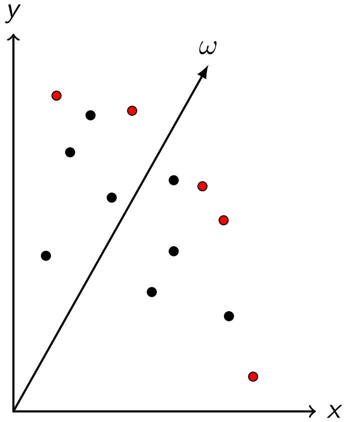
Regret Minimizing Set (RMS)



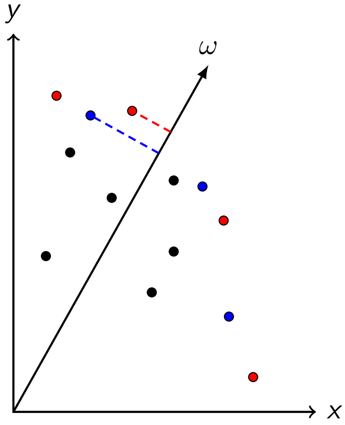
Regret Minimizing Set (RMS)



Regret Minimizing Set (RMS)



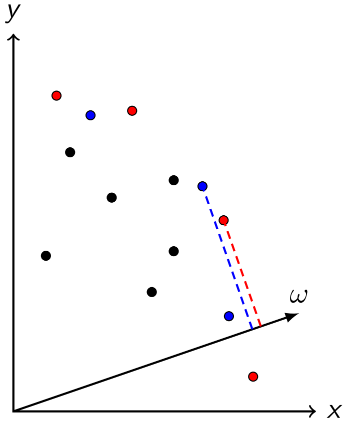
Regret Minimizing Set (RMS)



[NSL⁺10]:

$$\theta = \max_{|R|=r} \inf_{\omega \in \mathbb{R}^d_+} \frac{\max_{p \in R} \langle p, \omega \rangle}{\max_{p \in D} \langle p, \omega \rangle}$$

Regret Minimizing Set (RMS)



[NSL+10]:

$$\theta = \max_{|R|=r} \inf_{\omega \in \mathbb{R}_+^d} \frac{\max_{p \in R} \langle p, \omega \rangle}{\max_{p \in D} \langle p, \omega \rangle}$$

k-Regret Minimizing Set (k-RMS)

[CTVW14]:

$$\theta = \max_{|R|=r} \inf_{\omega \in \mathbb{R}^d_+} \frac{\max_{p \in R} \langle p, \omega \rangle}{\max^{(k)}_{p \in D} \langle p, \omega \rangle}.$$

k-Regret Minimizing Set (k-RMS)

[CTVW14]:

$$\theta = \max_{|R|=r} \inf_{\omega \in \mathbb{R}^d_+} \frac{\max_{p \in R} \langle p, \omega \rangle}{\max^{(k)}_{p \in D} \langle p, \omega \rangle}. \quad (\theta \leq 1)$$

k-Regret Minimizing Set (k-RMS)

[CTVW14]:

$$\theta = \max_{|R|=r} \inf_{\omega \in \mathbb{R}^d_+} \frac{\max_{p \in R} \langle p, \omega \rangle}{\max^{(k)}_{p \in D} \langle p, \omega \rangle}. \quad (\theta \leq 1)$$

- **Optimization**:
  Given D, r, k, find out the optimal R.

k-Regret Minimizing Set (k-RMS)

[CTVW14]:

$$\theta = \max_{|R|=r} \inf_{\omega \in \mathbb{R}^d_+} \frac{\max_{p\in R} \langle p, \omega \rangle}{\max^{(k)}_{p \in D} \langle p, \omega \rangle}. \quad (\theta \leq 1)$$

- **Optimization**:
  Given D, r, k, find out the optimal R.
- **Decision**:
  Given D, r, k, decide availability of θ.

Main Results

- When $d = 2$:

Main Results

- When $d = 2$:
  - Dec-k-RMS in $O(n + m)$ with preprocessing.
    ($n = |D|$ and m is the size of k-level set)

Main Results

- When $d = 2$:
  - Dec-k-RMS in $O(n + m)$ with preprocessing.
    ($n = |D|$ and m is the size of k-level set)
  - k-RMS in $O(n^2 \log n)$.
    - Previous result [CTVW14]: $O(rn^2 k^{1/3} + n^2 \log n)$.

Main Results

- When $d = 2$:
  - Dec-k-RMS in $O(n + m)$ with preprocessing.
    ($n = |D|$ and m is the size of k-level set)
  - k-RMS in $O(n^2 \log n)$.
    - Previous result [CTVW14]: $O(rn^2 k^{1/3} + n^2 \log n)$.

Main Results

- When $d = 2$:
  - Dec-k-RMS in $O(n + m)$ with preprocessing.
    ($n = |D|$ and m is the size of k-level set)
  - k-RMS in $O(n^2 \log n)$.
    - Previous result [CTVW14]: $O(rn^2 k^{1/3} + n^2 \log n)$.
  - RMS in *expected* $O(n \log n)$.

Main Results

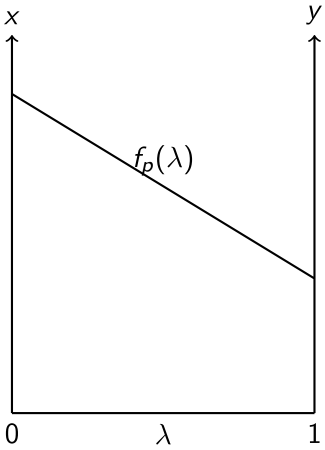
- When $d = 2$:
  - Dec-k-RMS in $O(n+m)$ with preprocessing.
    ($n = |D|$ and m is the size of k-level set)
  - k-RMS in $O(n^2 \log n)$.
    - Previous result [CTVW14]: $O(rn^2 k^{1/3} + n^2 \log n)$.
  - RMS in *expected* $O(n \log n)$.

Main Results

- When $d = 2$:
  - Dec-k-RMS in $O(n + m)$ with preprocessing. ($n = |D|$ and m is the size of k-level set)
  - k-RMS in $O(n^2 \log n)$.
    - Previous result [CTVW14]: $O(rn^2 k^{1/3} + n^2 \log n)$.
  - RMS in *expected* $O(n \log n)$.
- When $d \geq 3$, Dec-RMS is NP-hard.

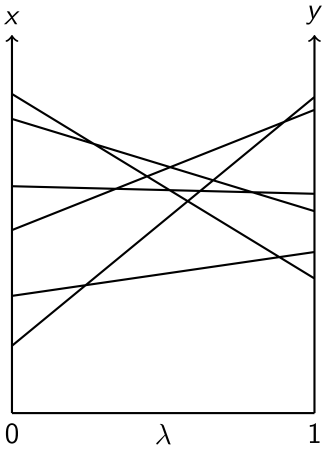
Geometric View

$$
\begin{aligned}
p &= (x, y) \\
&\Downarrow \\
f_p(\lambda) &= \langle p, \omega \rangle \\
&= (1-\lambda)x + \lambda y.
\end{aligned}
$$



Geometric View

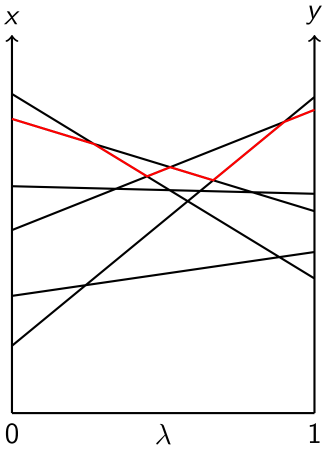
$$p = (x, y)$$
$$\Downarrow$$
$$f_p(\lambda) = \langle p, \omega \rangle$$
$$= (1 - \lambda)x + \lambda y.$$



Geometric View

$$p = (x, y)$$
$$\Downarrow$$
$$f_p(\lambda) = \langle p, \omega \rangle$$
$$= (1 - \lambda)x + \lambda y.$$

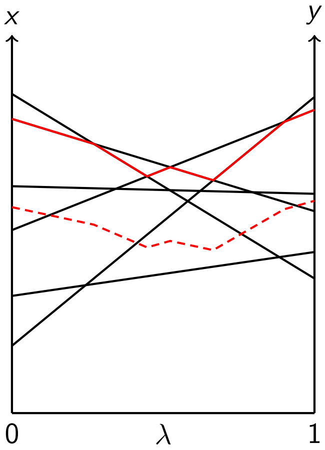
- k-level set (LS_k)



Geometric View

$$p = (x, y)$$
$$\Downarrow$$
$$f_p(\lambda) = \langle p, \omega \rangle$$
$$= (1 - \lambda)x + \lambda y.$$

- k-level set (LS_k)
- Scaled k-level set (θ-LS_k)

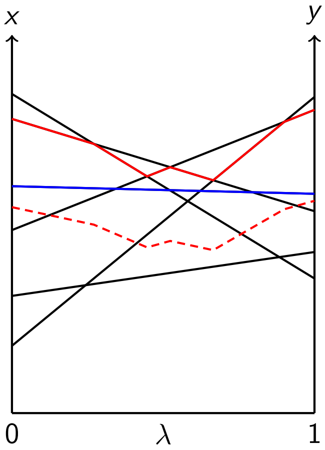


Geometric View

$$p = (x, y)$$
$$\Downarrow$$
$$\begin{aligned} f_p(\lambda) &= \langle p, \omega \rangle \\ &= (1-\lambda)x + \lambda y. \end{aligned}$$

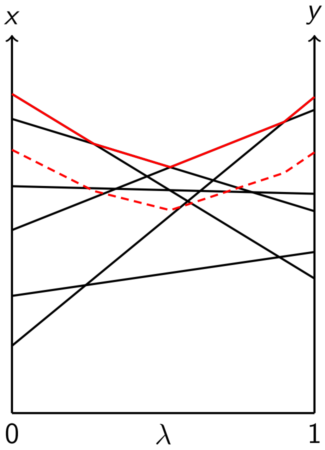
- k-level set (LS_k)
- Scaled k-level set $(\theta\text{-}LS_k)$

$$\max_{p \in R} f_p(\lambda) \geq \theta\text{-}LS_k(\lambda)$$



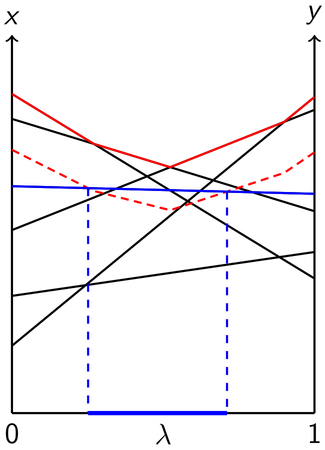
Warm-Up: Dec-RMS

- When $k = 1$, LS_1 (and thus θ-LS_1) is convex.



Warm-Up: Dec-RMS

- When $k = 1$, LS_1 (and thus θ-LS_1) is convex.
- Reduced to interval cover.



Dec-k-RMS

- **Goal**: Cover polygonal $C(\lambda)$ with a bunch of lines $f_p(\lambda)$.

Dec-k-RMS

- **Goal**: Cover polygonal $C(\lambda)$ with a bunch of lines $f_p(\lambda)$.
- **Problem**:
  The intervals that f_p covers C may be *disconnected*.

Dec-k-RMS

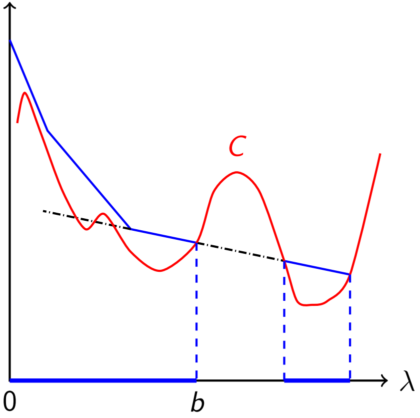
- **Goal**: Cover polygonal $C(\lambda)$ with a bunch of lines $f_p(\lambda)$.
- **Problem**:
  The intervals that f_p covers C may be *disconnected*.
- **Solution**:
  Greedily increment the initial covered interval $[0, b]$.

Dec-k-RMS

- Suppose records are sorted:
  $x_1 \geq \ldots \geq x_n \geq 0, 0 \leq y_1 \leq \ldots \leq y_n$.

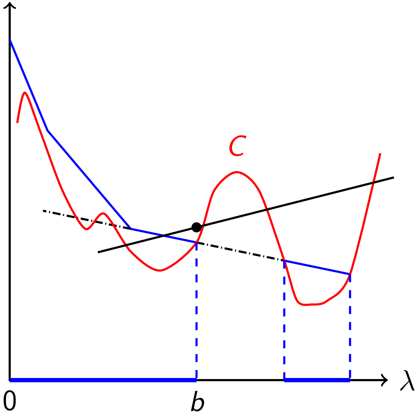
Dec-k-RMS

- Suppose records are sorted:
  $x_1 \geq \ldots \geq x_n \geq 0, 0 \leq y_1 \leq \ldots \leq y_n$.
- Then the rest covered intervals disconnected from the initial one are *useless*:



Dec-k-RMS

- Suppose records are sorted:
  $x_1 \geq \ldots \geq x_n \geq 0, 0 \leq y_1 \leq \ldots \leq y_n$.
- Then the rest covered intervals disconnected from the initial one are *useless*:



Dec-k-RMS

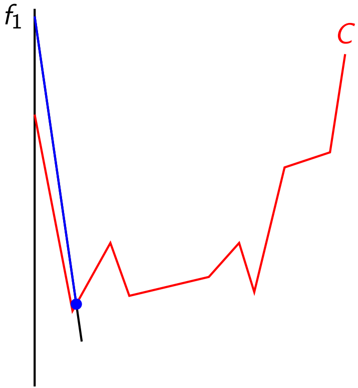
We apply the following greedy algorithm:

- Keep selected lines R as a stack;
- Once the new line increments the initial covered interval, push it into R;
- While pushing, pop out the redundant lines from R.

Dec-k-RMS (Example)

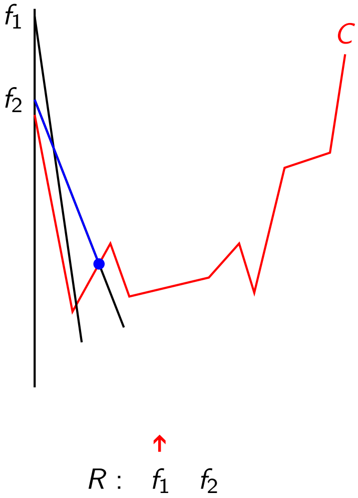
R :

Dec-k-RMS (Example)

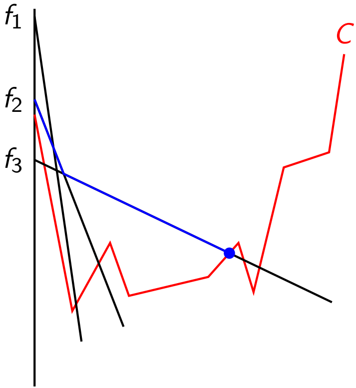


$R: \quad f_1$

Dec-k-RMS (Example)

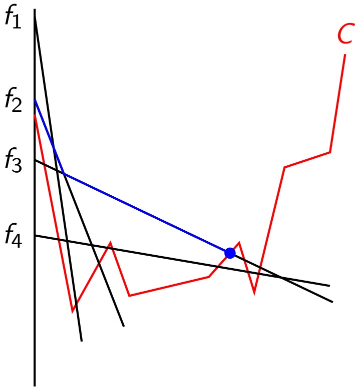


Dec-k-RMS (Example)



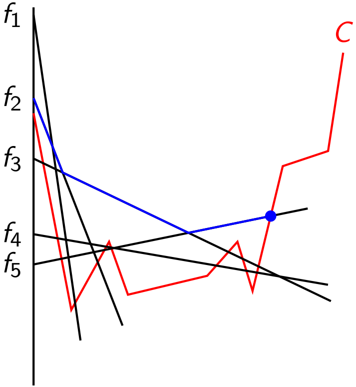
$R: \quad f_2 \quad f_3$

Dec-k-RMS (Example)



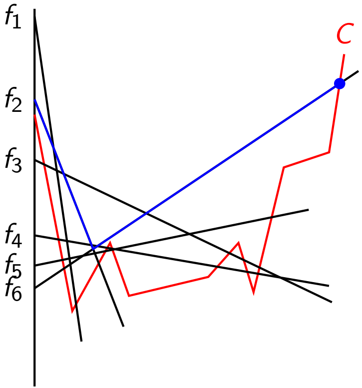
$R: \quad f_2 \quad f_3 \quad f_4$

Dec-k-RMS (Example)



$R: \quad f_2 \quad f_3 \quad f_5$

Dec-k-RMS (Example)



$R: \quad f_2 \quad f_3 \quad f_5 \quad f_6$

Dec-k-RMS (Data Structure)

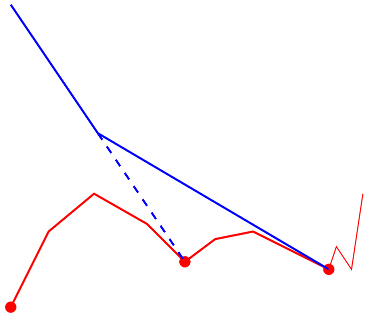
- b_i = the value of b when f_i is pushed in.

Dec-k-RMS (Data Structure)

- b_i = the value of b when f_i is pushed in.
- Suppose at a moment $R = (f_{i_1}, \ldots, f_{i_r})$; then we store the **convex hull** of $C(\lambda)$ for $\lambda \in [b_{i_\ell}, b_{i_{\ell+1}}]$.

Dec-k-RMS (Data Structure)

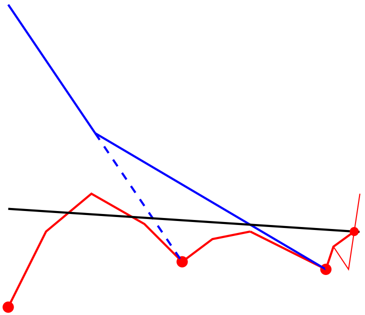
- b_i = the value of b when f_i is pushed in.
- Suppose at a moment $R = (f_{i_1}, \ldots, f_{i_r})$; then we store the **convex hull** of $C(\lambda)$ for $\lambda \in [b_{i_\ell}, b_{i_{\ell+1}}]$.
- The stack of convex hulls need to support the following operations:



Dec-k-RMS (Data Structure)

- b_i = the value of b when f_i is pushed in.
- Suppose at a moment $R = (f_{i_1}, \ldots, f_{i_r})$; then we store the **convex hull** of $C(\lambda)$ for $\lambda \in [b_{i_\ell}, b_{i_{\ell+1}}]$.
- The stack of convex hulls need to support the following operations:

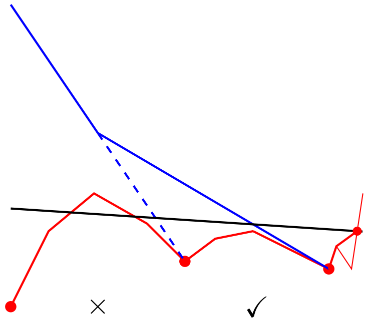
Create



Dec-k-RMS (Data Structure)

- b_i = the value of b when f_i is pushed in.
- Suppose at a moment $R = (f_{i_1}, \ldots, f_{i_r})$; then we store the **convex hull** of $C(\lambda)$ for $\lambda \in [b_{i_\ell}, b_{i_{\ell+1}}]$.
- The stack of convex hulls need to support the following operations:

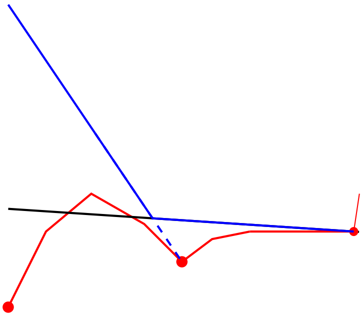
Test coverage



Dec-k-RMS (Data Structure)

- b_i = the value of b when f_i is pushed in.
- Suppose at a moment $R = (f_{i_1}, \ldots, f_{i_r})$; then we store the **convex hull** of $C(\lambda)$ for $\lambda \in [b_{i_\ell}, b_{i_{\ell+1}}]$.
- The stack of convex hulls need to support the following operations:

Merge



Dec-k-RMS (Time Analysis)

- m: the size of (number of segments in) LS_k.

Dec-k-RMS (Time Analysis)

- m: the size of (number of segments in) LS_k.
- Calculate LS_k: $O(n \log m + m \log^{1+\delta} k)$ ([Cha99]);

Dec-k-RMS (Time Analysis)

- m: the size of (number of segments in) LS_k.
- Calculate LS_k: $O(n \log m + m \log^{1+\delta} k)$ ([Cha99]);
- Sort the lines: $O(n \log n)$;

Dec-k-RMS (Time Analysis)

- m: the size of (number of segments in) LS_k.
- Calculate LS_k: $O(n \log m + m \log^{1+\delta} k)$ ([Cha99]);
- Sort the lines: $O(n \log n)$;
- Maintain the stack of convex hulls: $O(n + m)$.

k-RMS

- **Goal**: binary search over θ, and call the algorithm of Dec-k-RMS.

k-RMS

- **Goal**: binary search over θ, and call the algorithm of Dec-k-RMS.
- All possible values of θ:

$$Cand(D) := \left\{ \frac{f_p(\lambda)}{LS_k(\lambda)} \mid p \in D, \lambda \in X(D) \right\}.$$

k-RMS

- **Goal**: binary search over θ, and call the algorithm of Dec-k-RMS.
- All possible values of θ:

$$Cand(D) := \left\{ \frac{f_p(\lambda)}{LS_k(\lambda)} \mid p \in D, \lambda \in X(D) \right\}.$$

- **Problem**: $|Cand(D)|$ can be as large as $\Theta(n^3)$.

k-RMS

- **Goal**: binary search over θ, and call the algorithm of Dec-k-RMS.
- All possible values of θ:

$$Cand(D) := \left\{ \frac{f_p(\lambda)}{LS_k(\lambda)} \mid p \in D, \lambda \in X(D) \right\}.$$

- **Problem**: $|Cand(D)|$ can be as large as $\Theta(n^3)$.
- **Solution**: Implicitly store $|Cand(D)|$ using *sweep line* over $X(D)$.

k-RMS

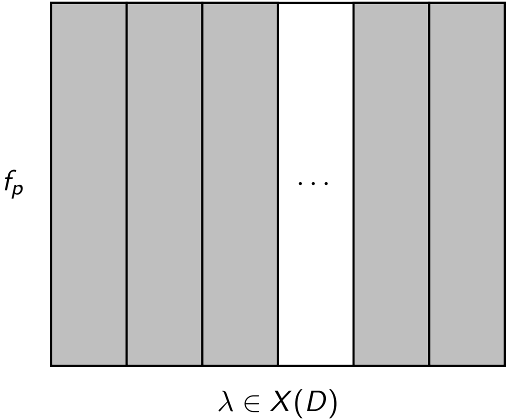
- We can access j_λ-th largest $f_p(\lambda)$ for each $\lambda \in X(D)$ once in $|X(D)| = O(n^2)$ time.

k-RMS

- We can access j_λ-th largest $f_p(\lambda)$ for each $\lambda \in X(D)$ once in $|X(D)| = O(n^2)$ time.
- Maintain a search range for every $\lambda \in X(D)$. Half of them are reduced by half in each round.

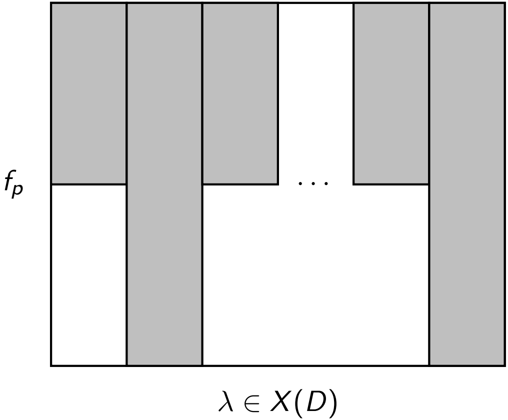
k-RMS

- We can access j_λ-th largest $f_p(\lambda)$ for each $\lambda \in X(D)$ once in $|X(D)| = O(n^2)$ time.
- Maintain a search range for every $\lambda \in X(D)$. Half of them are reduced by half in each round.



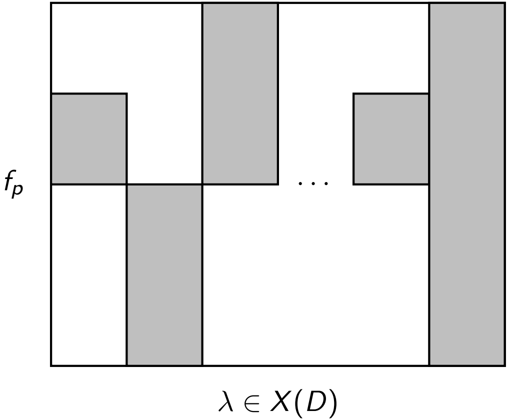
k-RMS

- We can access j_λ-th largest $f_p(\lambda)$ for each $\lambda \in X(D)$ once in $|X(D)| = O(n^2)$ time.
- Maintain a search range for every $\lambda \in X(D)$. Half of them are reduced by half in each round.



k-RMS

- We can access j_λ-th largest $f_p(\lambda)$ for each $\lambda \in X(D)$ once in $|X(D)| = O(n^2)$ time.
- Maintain a search range for every $\lambda \in X(D)$. Half of them are reduced by half in each round.



k-RMS

- Using **weighted median**, we can ensure a stable $1/4$-reduction of candidate values.

k-RMS

- Using **weighted median**, we can ensure a stable 1/4-reduction of candidate values.
- $O(n^2)$ time each round, $O(\log n)$ rounds.
- $O(n \log m + m \log^{1+\delta} k) = o(n^2)$ preprocessing.

RMS

- All possible values of θ:

$$Cand(D) := \left\{ \frac{f_p(\lambda)}{LS_1(\lambda)} \mid p, q \in D,\ f_p(\lambda) = f_q(\lambda) \right\}$$

correspond to intersection points.

RMS

- All possible values of θ:

$$Cand(D) := \left\{ \frac{f_p(\lambda)}{LS_1(\lambda)} \mid p, q \in D,\ f_p(\lambda) = f_q(\lambda) \right\}$$

  correspond to intersection points.
- For a search range $[\theta_0, \theta_1]$, the candidate values could not be **listed**, but could be **random accessed**.

RMS

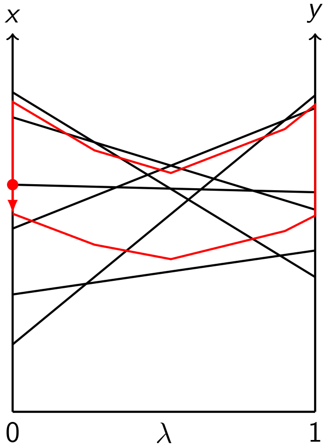
- All possible values of θ:

$$Cand(D) := \left\{ \frac{f_p(\lambda)}{LS_1(\lambda)} \mid p, q \in D,\ f_p(\lambda) = f_q(\lambda) \right\}$$

  correspond to intersection points.
- For a search range $[\theta_0, \theta_1]$, the candidate values could not be **listed**, but could be **random accessed**.
- $O(n \log n)$ time for n random sampling each round, expected $O(1)$ round.

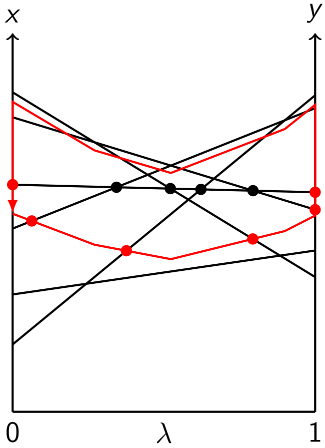
RMS

Traverse the $O(n)$ endpoints on the boundary in counterclockwise:



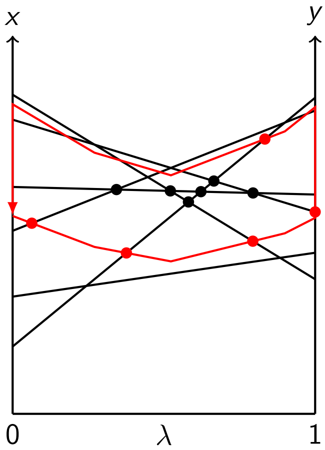
RMS

Traverse the $O(n)$ endpoints on the boundary in counterclockwise:



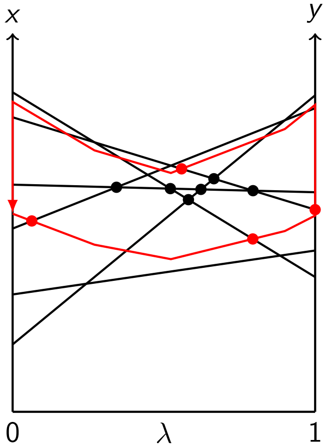
RMS

Traverse the $O(n)$ endpoints on the boundary in counterclockwise:



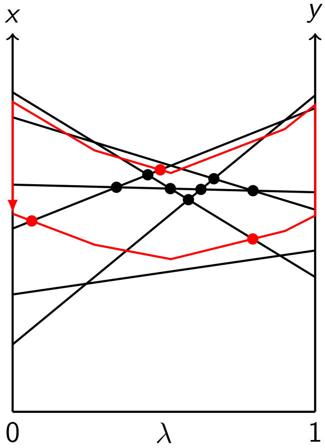
RMS

Traverse the $O(n)$ endpoints on the boundary in counterclockwise:



RMS

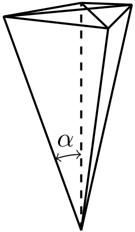
Traverse the $O(n)$ endpoints on the boundary in counterclockwise:



3D Dec-RMS is NP-hard

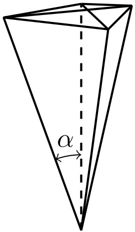
- **Observation**: If a spherical triangle $\triangle ABC$ contains the circumcenter P, then $\{A, B, C\}$ has fixed regret ratio in respect to P:

$$1 - \theta = 1 - \cos\alpha.$$



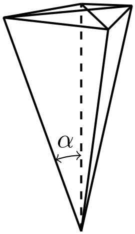
3D Dec-RMS is NP-hard

- **Observation**: If a spherical triangle $\triangle ABC$ contains the circumcenter P, then $\{A, B, C\}$ has fixed regret ratio in respect to P:
$$1 - \theta = 1 - \cos\alpha.$$
- So for $\theta \approx \cos\alpha$, one either choose P or choose ABC.



3D Dec-RMS is NP-hard

- **Observation**: If a spherical triangle $\triangle ABC$ contains the circumcenter P, then $\{A, B, C\}$ has fixed regret ratio in respect to P:
$$1 - \theta = 1 - \cos\alpha.$$
- So for $\theta \approx \cos\alpha$, one either choose P or choose ABC.
- Reduced to **Vertex Cover** on a highly constraint class of planar graphs, which we project to the sphere surface.



Timothy M Chan.
Remarks on k-level algorithms in the plane.
Manuscript, Department of Computer Science, University of Waterloo, Waterloo, Canada, 1999.

Sean Chester, Alex Thomo, S Venkatesh, and Sue Whitesides.
Computing k-regret minimizing sets.
Proceedings of VLDB, 7(5), 2014.

Danupon Nanongkai, Atish Das Sarma, Ashwin Lall, Richard J Lipton, and Jun Xu.
Regret-minimizing representative databases.
Proceedings of the VLDB Endowment, 3(1-2):1114–1124, 2010.