Online Learning via Convex Geometry, with Applications to Pricing

Adrian Vladu
MIT

Dynamic Pricing Problem


$0.5
$2.0
$1.0

day 1


x 1


x 3


x 0


?

day 1


x 1


x 3


x 0


?

$7

day 1


x 1


x 3


x 0


?

$7

Regret=6.5

day 1


x 1


x 3


x 0


?

$7

Regret=6.5

day 1

x 1

x 3

x 0

?

$7

day 2

x 2

x 0

x 2

?

Regret=6.5

day 1

x 1

x 3

x 0

?

$7

day 2

x 2

x 0

x 2

?

$2

Regret=7.5

day 1

x 1

x 3

x 0

?

$7

day 2

x 2

x 0

x 2

?

$2

$0.5

$2.0

$1.0

Regret=7.5

day 1

x 1

x 3

x 0

?

$7

day 2

x 2

x 0

x 2

?

$2

day 3

x 0

x 1

x 1

?

Regret=7.5

day 1

x 1

x 3

x 0

?

$7

day 2

x 2

x 0

x 2

?

$2

day 3

x 0

x 1

x 1

?

$4

$0.5

$2.0

$1.0

Regret=10.5

day 1

x 1

x 3

x 0

?

$7

day 2

x 2

x 0

x 2

?

$2

day 3

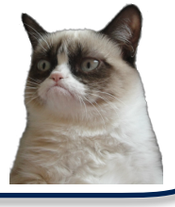
x 0

x 1

x 1

?

$4



Regret=10.5

day 1

x 1

x 3

x 0

?

$7

day 2

x 2

x 0

x 2

?

$2

day 3

x 0

x 1

x 1

?

$4

day 4

x 1

x 1

x 1

?

Regret=10.5

day 1

x 1

x 3

x 0

?

$7

❌

day 2

x 2

x 0

x 2

?

$2

✅

day 3

x 0

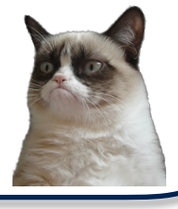
x 1

x 1

?

$4

❌



day 4

x 1

x 1

x 1

?

$3

Regret=11

day 1

x 1

x 3

x 0

?

$7

❌

day 2

x 2

x 0

x 2

?

$2

✅

day 3

x 0

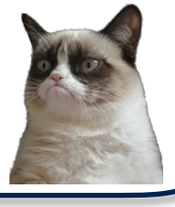
x 1

x 1

?

$4

❌



day 4

x 1

x 1

x 1

?

$3

✅

Dynamic Pricing Problem

- Warm-up: 1-dimensional case

Dynamic Pricing Problem

- Warm-up: 1-dimensional case

Dynamic Pricing Problem

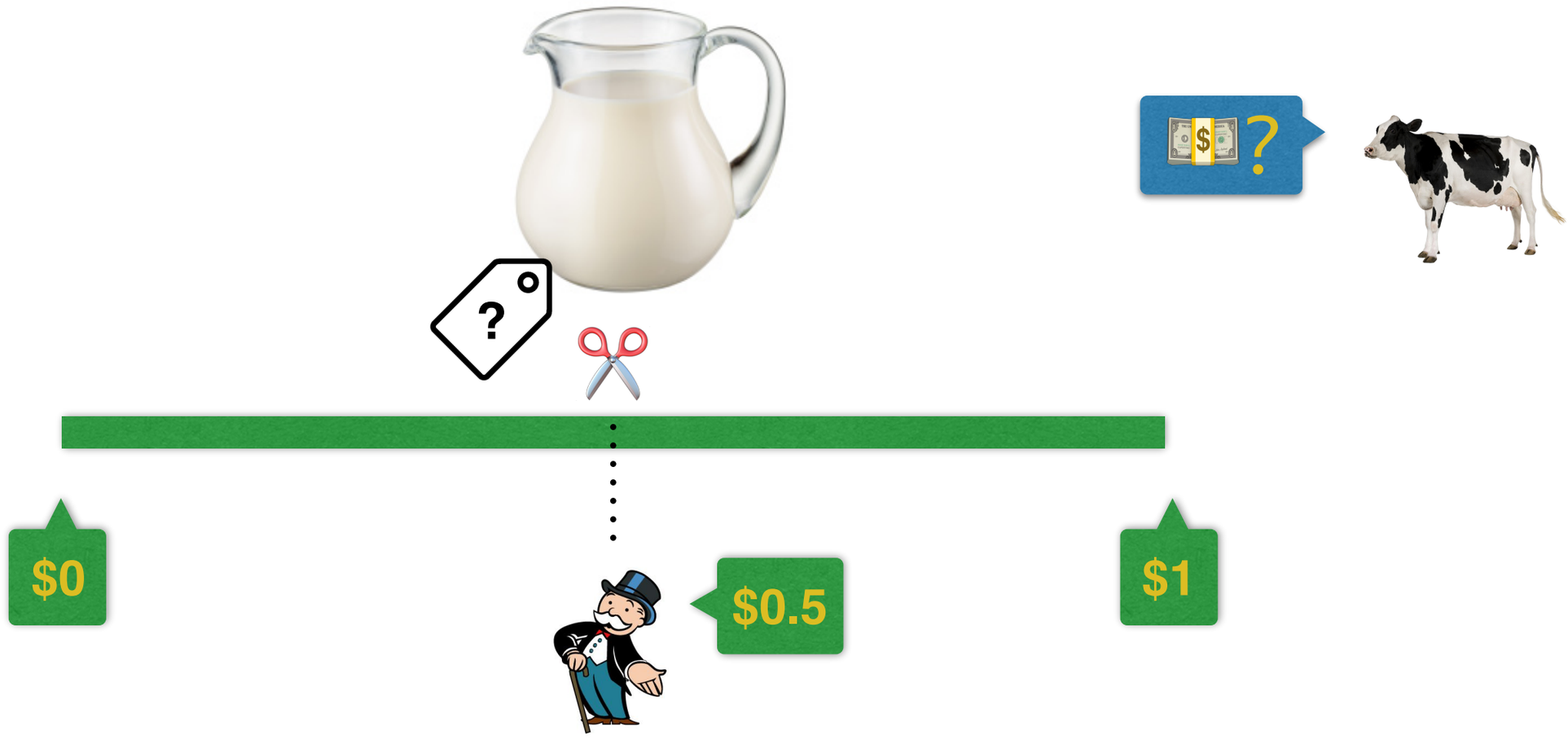
- Warm-up: 1-dimensional case

Dynamic Pricing Problem

- Warm-up: 1-dimensional case

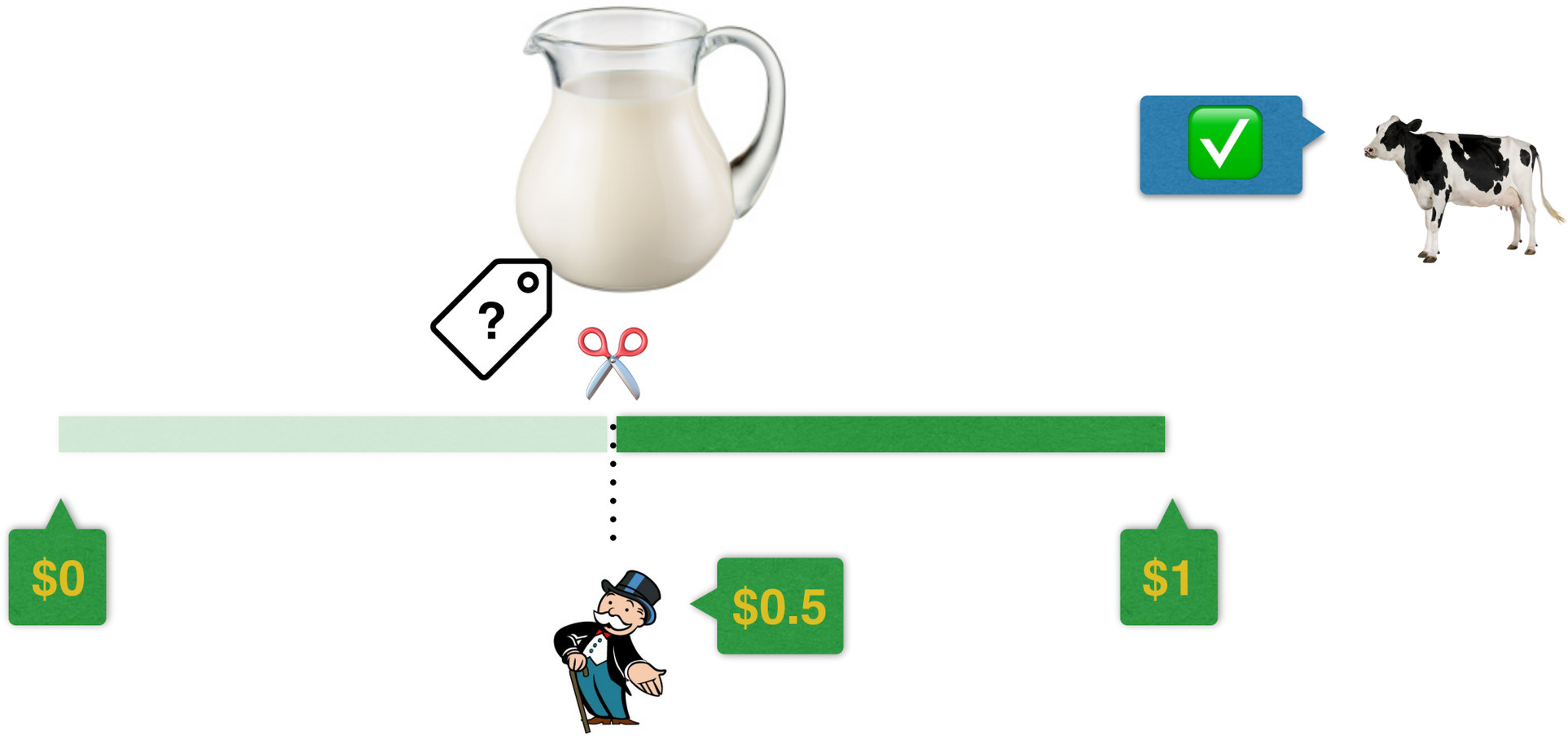
Dynamic Pricing Problem

- Warm-up: 1-dimensional case



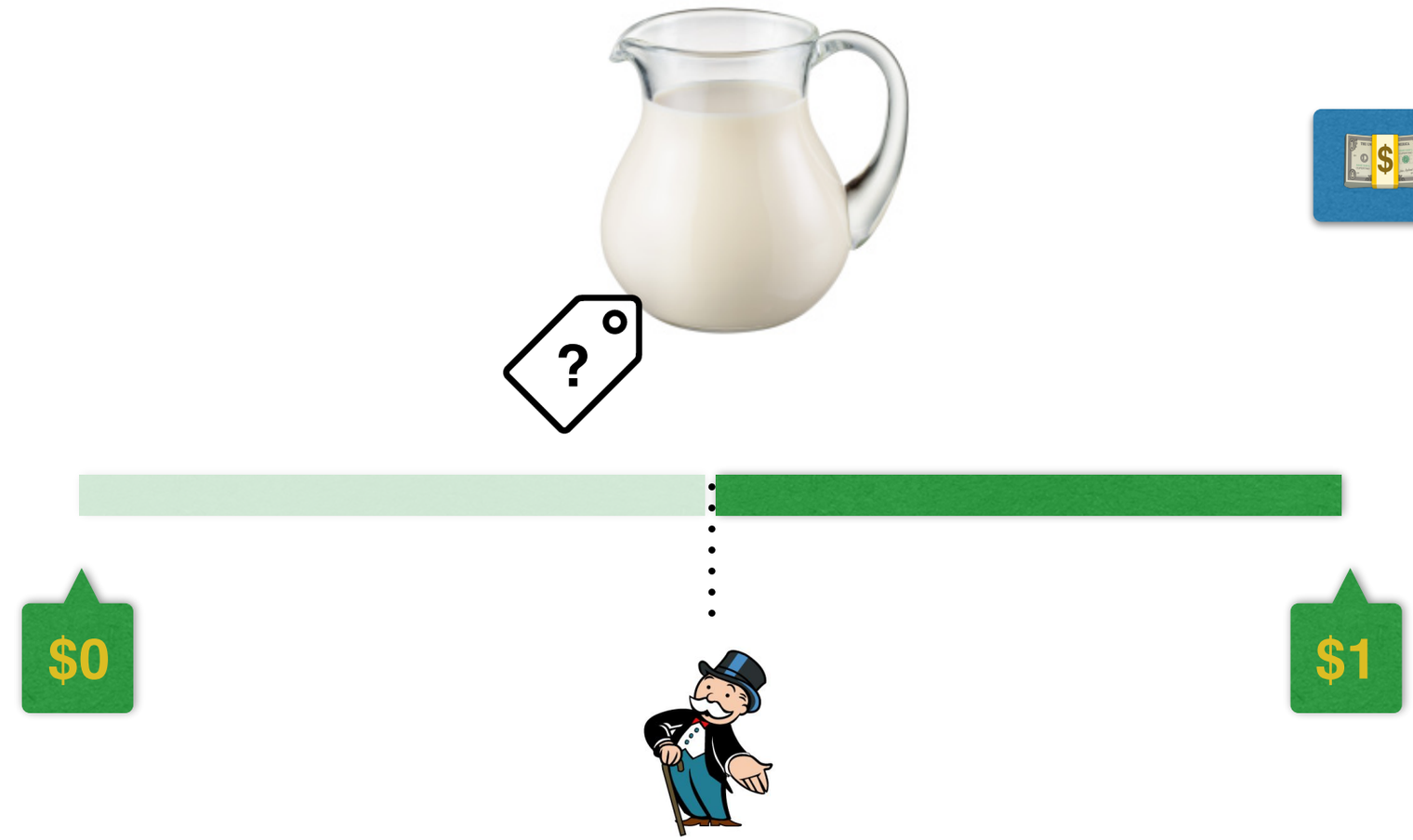
Dynamic Pricing Problem

- Warm-up: 1-dimensional case



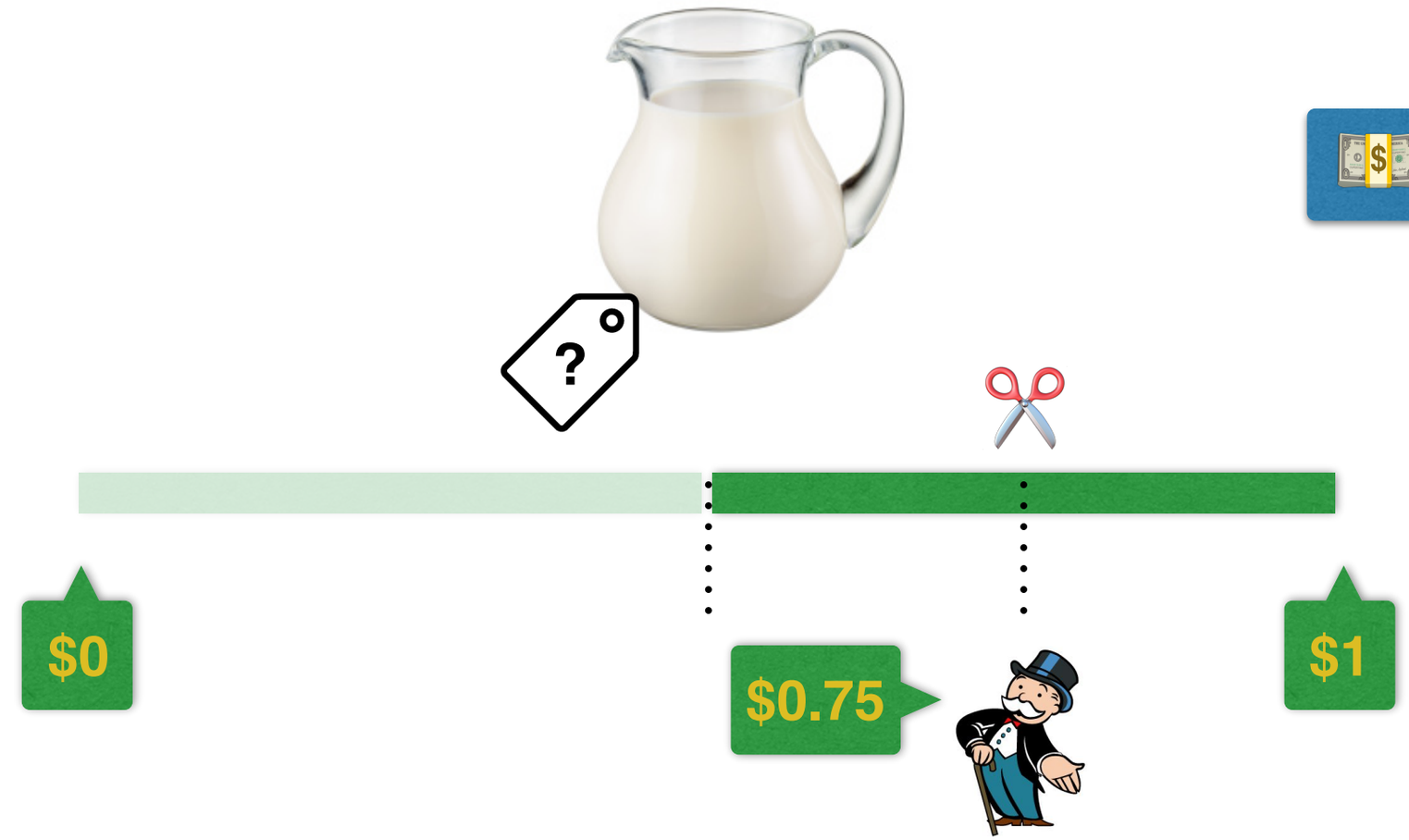
Dynamic Pricing Problem

- Warm-up: 1-dimensional case



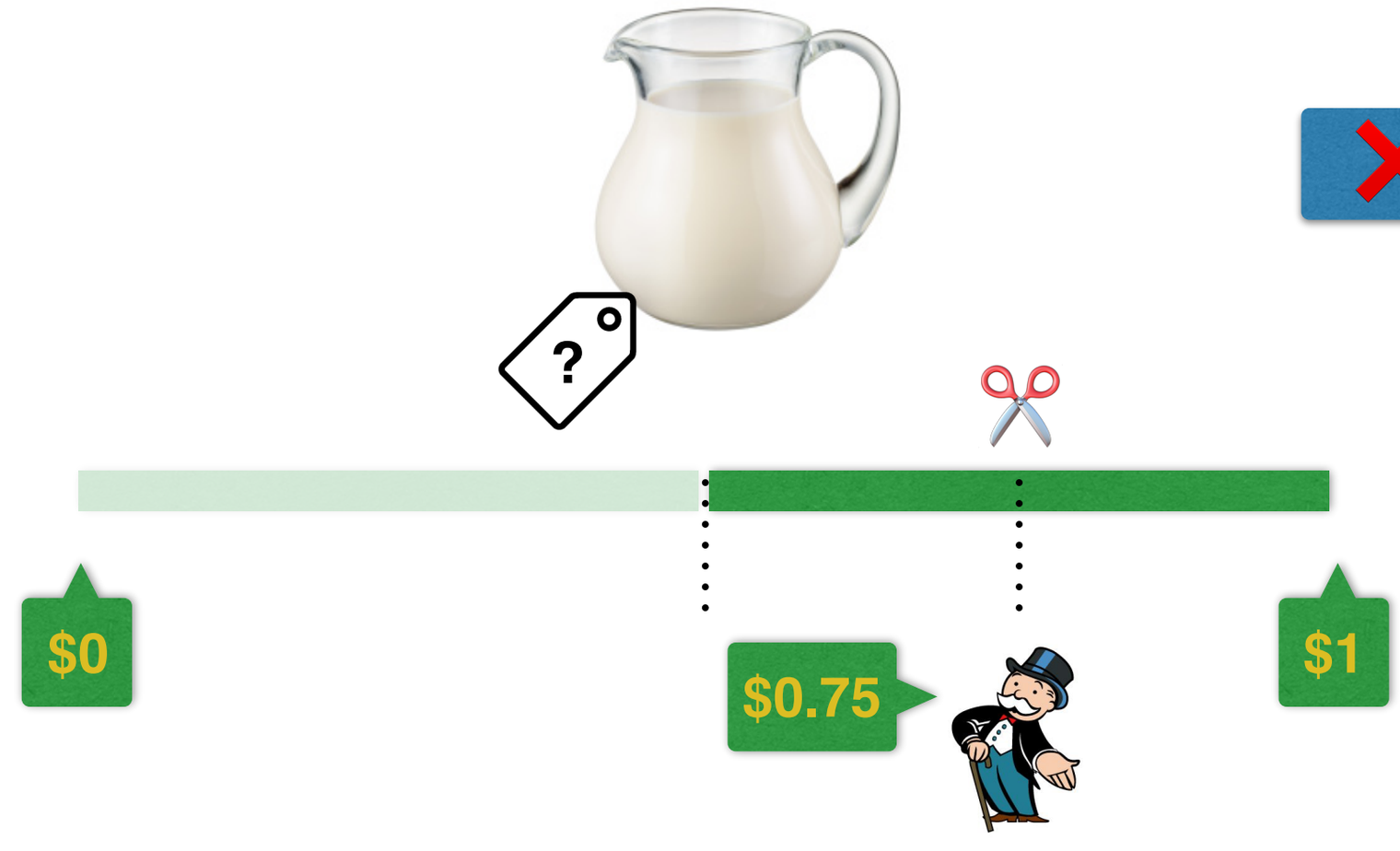
Dynamic Pricing Problem

- Warm-up: 1-dimensional case



Dynamic Pricing Problem

- Warm-up: 1-dimensional case

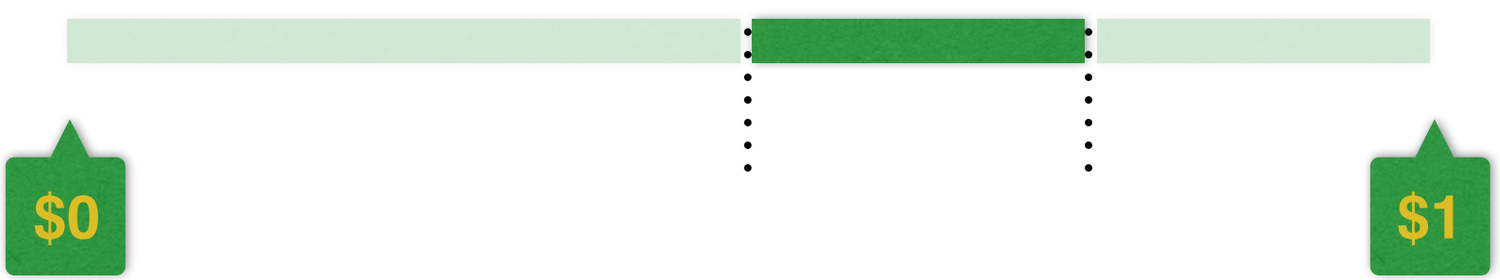


Dynamic Pricing Problem

- Warm-up: 1-dimensional case

Dynamic Pricing Problem

- Warm-up: 1-dimensional case

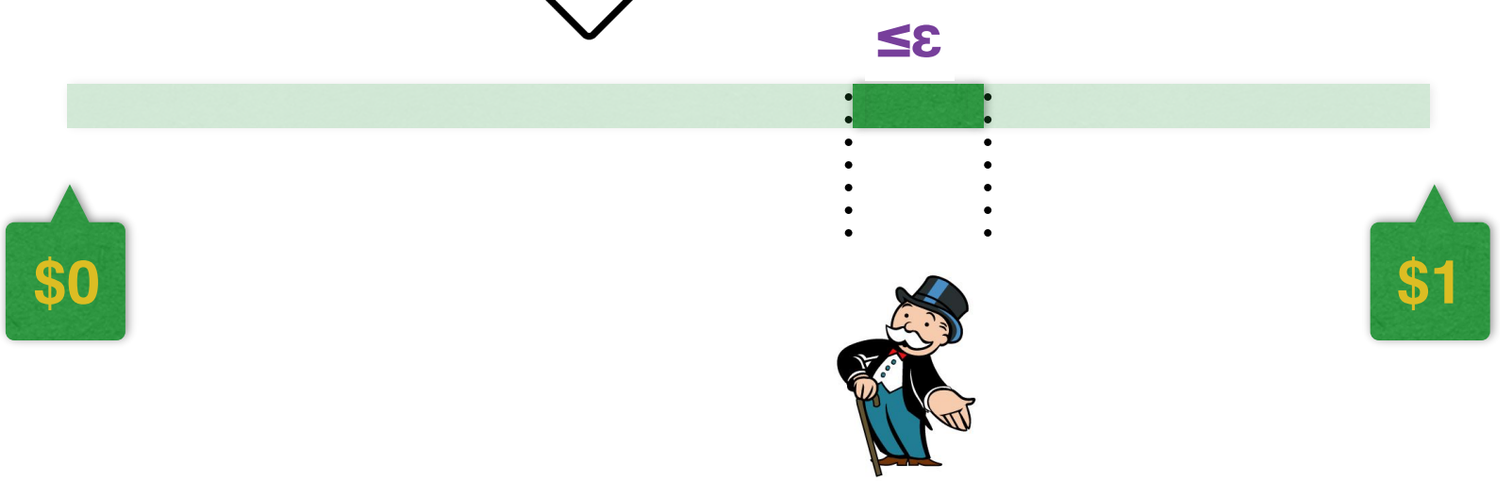


O(log 1/ε) days

Dynamic Pricing Problem

- Warm-up: 1-dimensional case

O(log 1/ε) days



Dynamic Pricing Problem

- Warm-up: 1-dimensional case
- **O(log T)** is easy

O(log 1/ε) days

Dynamic Pricing Problem

- Warm-up: 1-dimensional case
  - **O(log T)** is easy

Dynamic Pricing Problem

- Warm-up: 1-dimensional case
  - **O(log T)** is easy
  - **O(log log T)** is a cute puzzle **[Kleinberg, Leighton ‘03]**

Dynamic Pricing Problem

- Warm-up: 1-dimensional case
  - **O(log T)** is easy
  - **O(log log T)** is a cute puzzle **[Kleinberg, Leighton ‘03]**
- Back to d-dimensional case

Dynamic Pricing Problem

- Warm-up: 1-dimensional case
  - **O(log T)** is easy
  - **O(log log T)** is a cute puzzle **[Kleinberg, Leighton ‘03]**
- Back to d-dimensional case
  - **$O(d^2 \log T)$ [Cohen, Lobel, Paes Leme ’16]**

Dynamic Pricing Problem

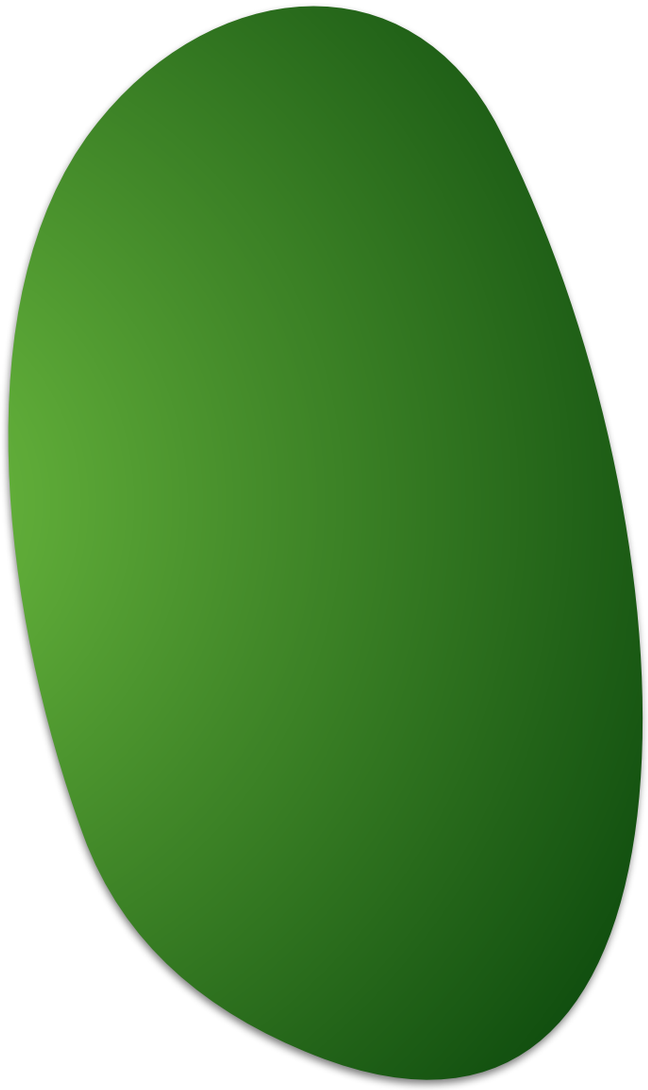
- Warm-up: 1-dimensional case
  - **O(log T)** is easy
  - **O(log log T)** is a cute puzzle **[Kleinberg, Leighton ‘03]**
- Back to d-dimensional case
  - **$O(d^2 \log T)$ [Cohen, Lobel, Paes Leme ’16]**
  - **$\Omega(d \log \log T)$** lower bound

Dynamic Pricing Problem

- Warm-up: 1-dimensional case
  - **O(log T)** is easy
  - **O(log log T)** is a cute puzzle **[Kleinberg, Leighton ‘03]**
- Back to d-dimensional case
  - **$O(d^2 \log T)$ [Cohen, Lobel, Paes Leme ’16]**
  - **$\Omega(d \log \log T)$** lower bound
  - Can we bridge the gap?

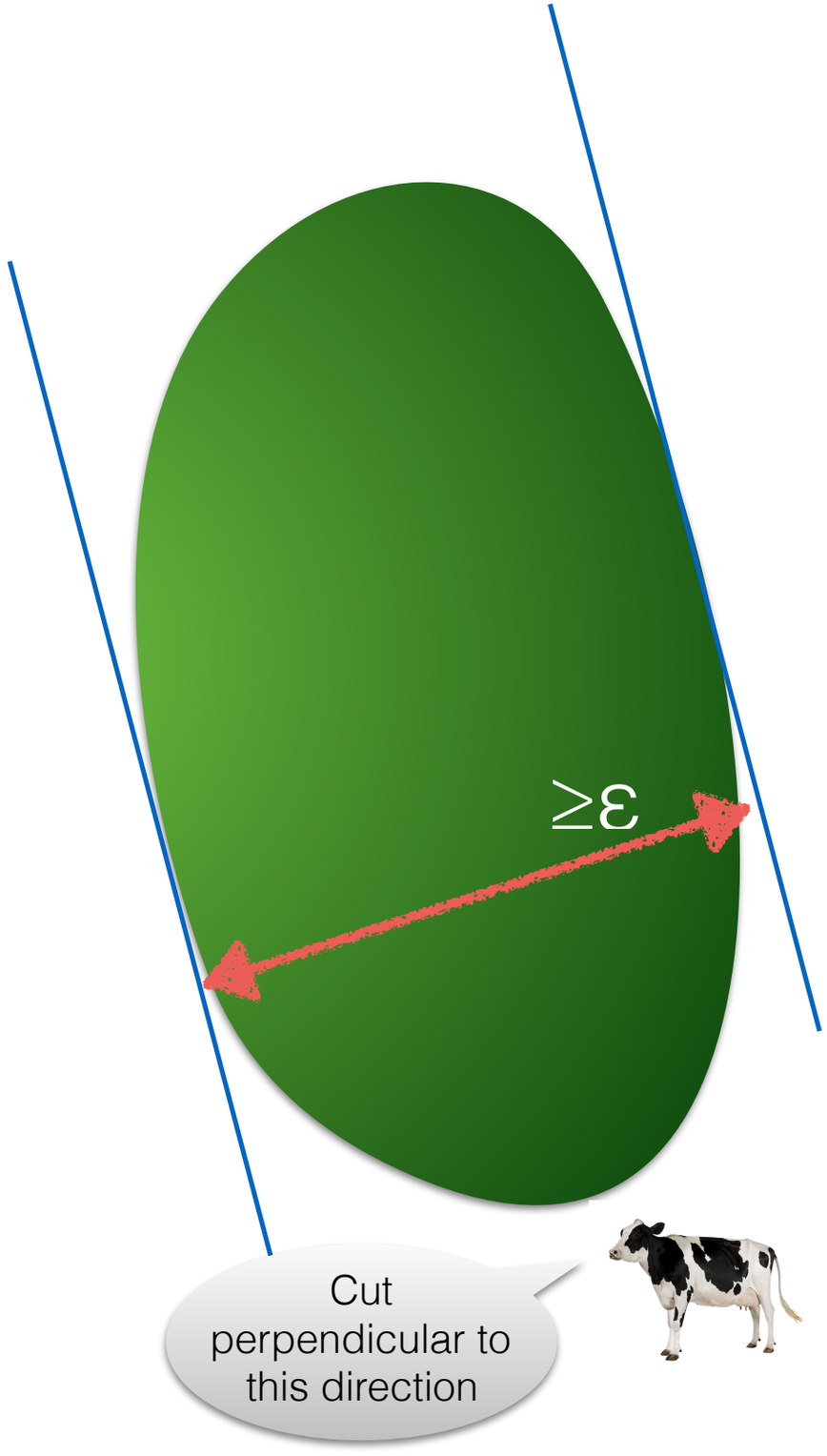
Dynamic Pricing Problem

- Warm-up: 1-dimensional case
  - **O(log T)** is easy
  - **O(log log T)** is a cute puzzle **[Kleinberg, Leighton ‘03]**
- Back to d-dimensional case
  - **$O(d^2 \log T)$ [Cohen, Lobel, Paes Leme ’16]**
  - **$\Omega(d \log \log T)$** lower bound
  - **O(d log T) [this work]**



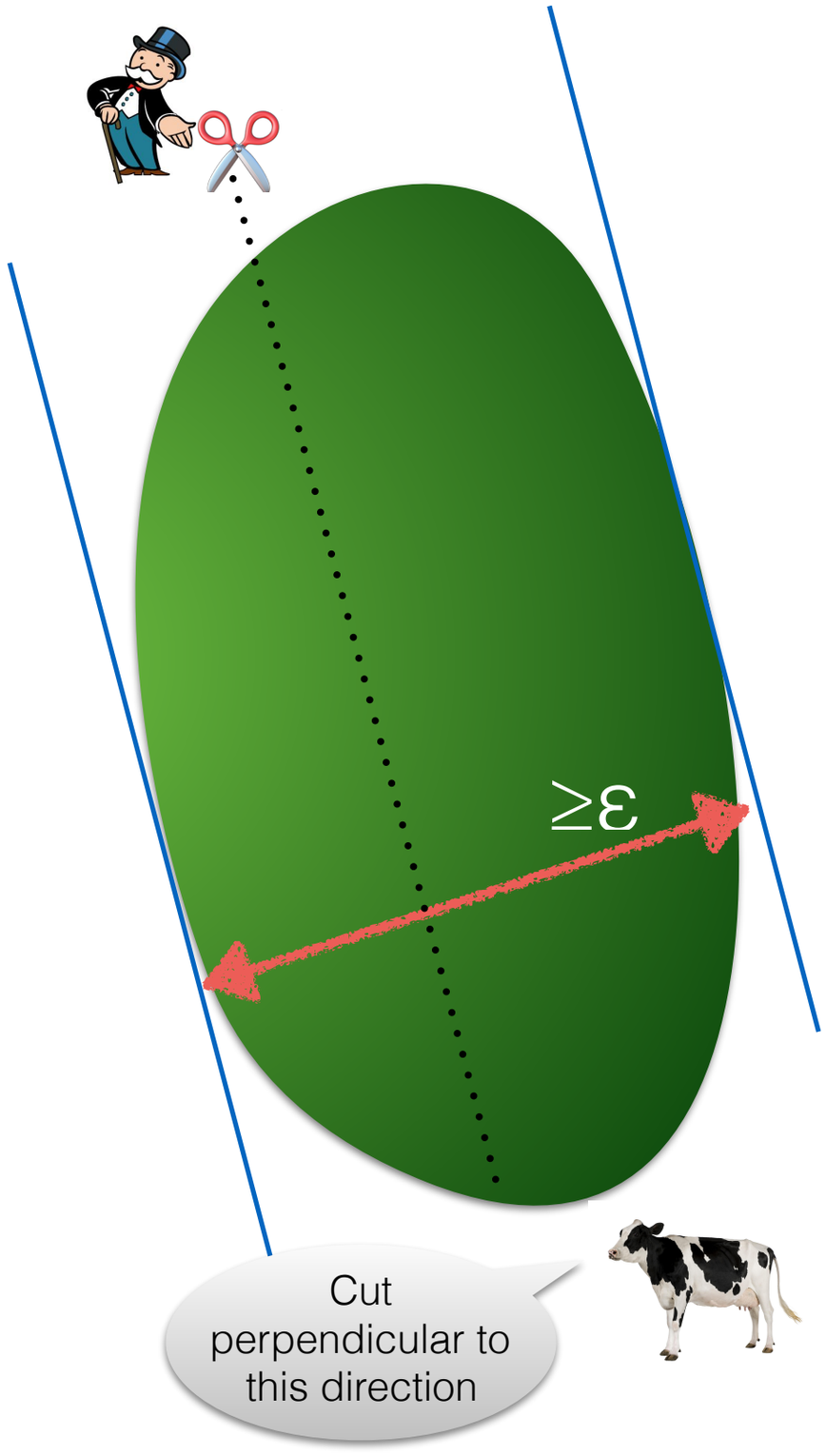
Model

- start with d-dimensional convex body (possible prices for items)



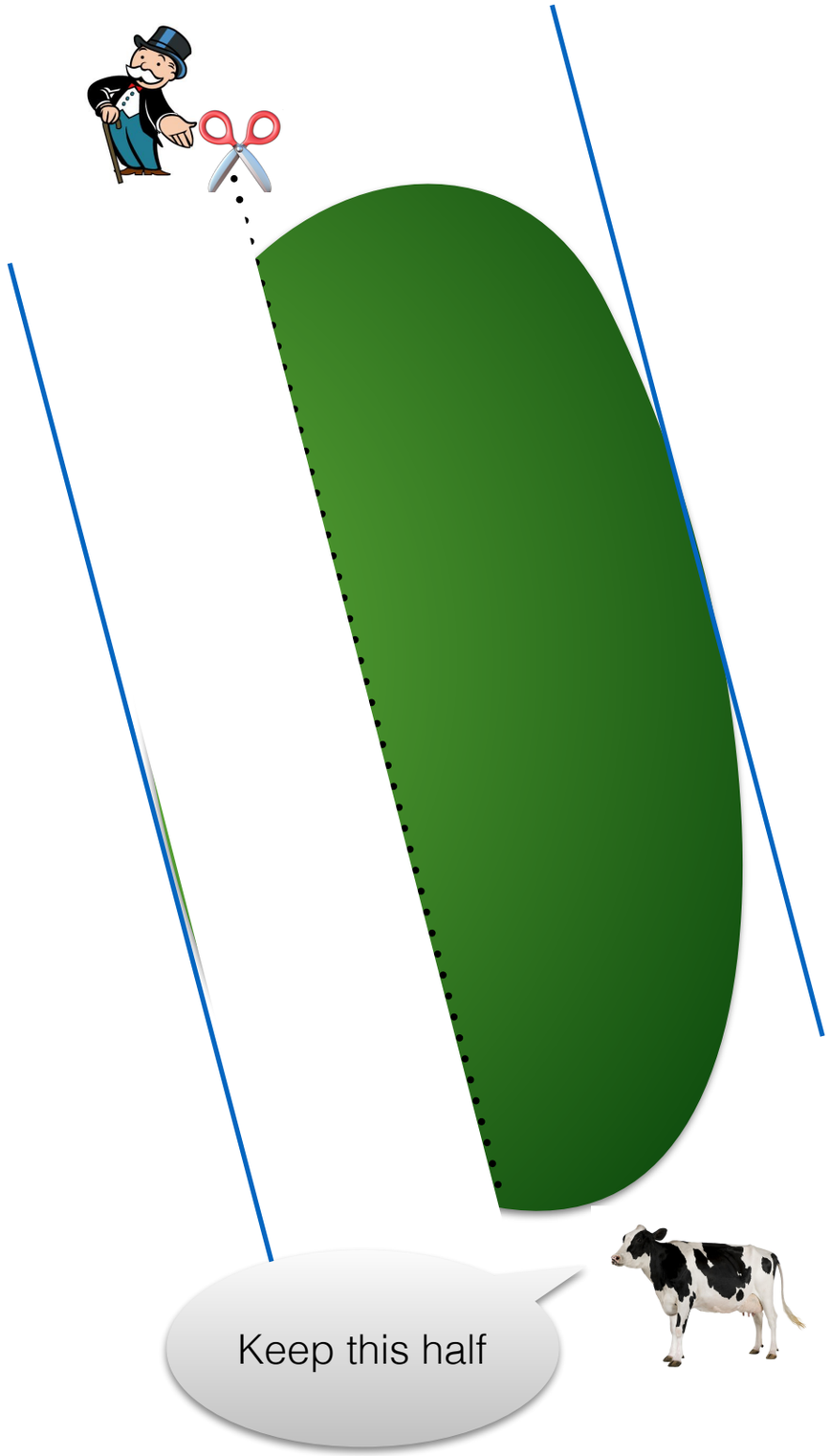
Model

- start with d-dimensional convex body (possible prices for items)
- customer gives vector of item quantities u, $\|u\| = 1$, with the promise that $w(K, u) \geq \epsilon$, asks for price



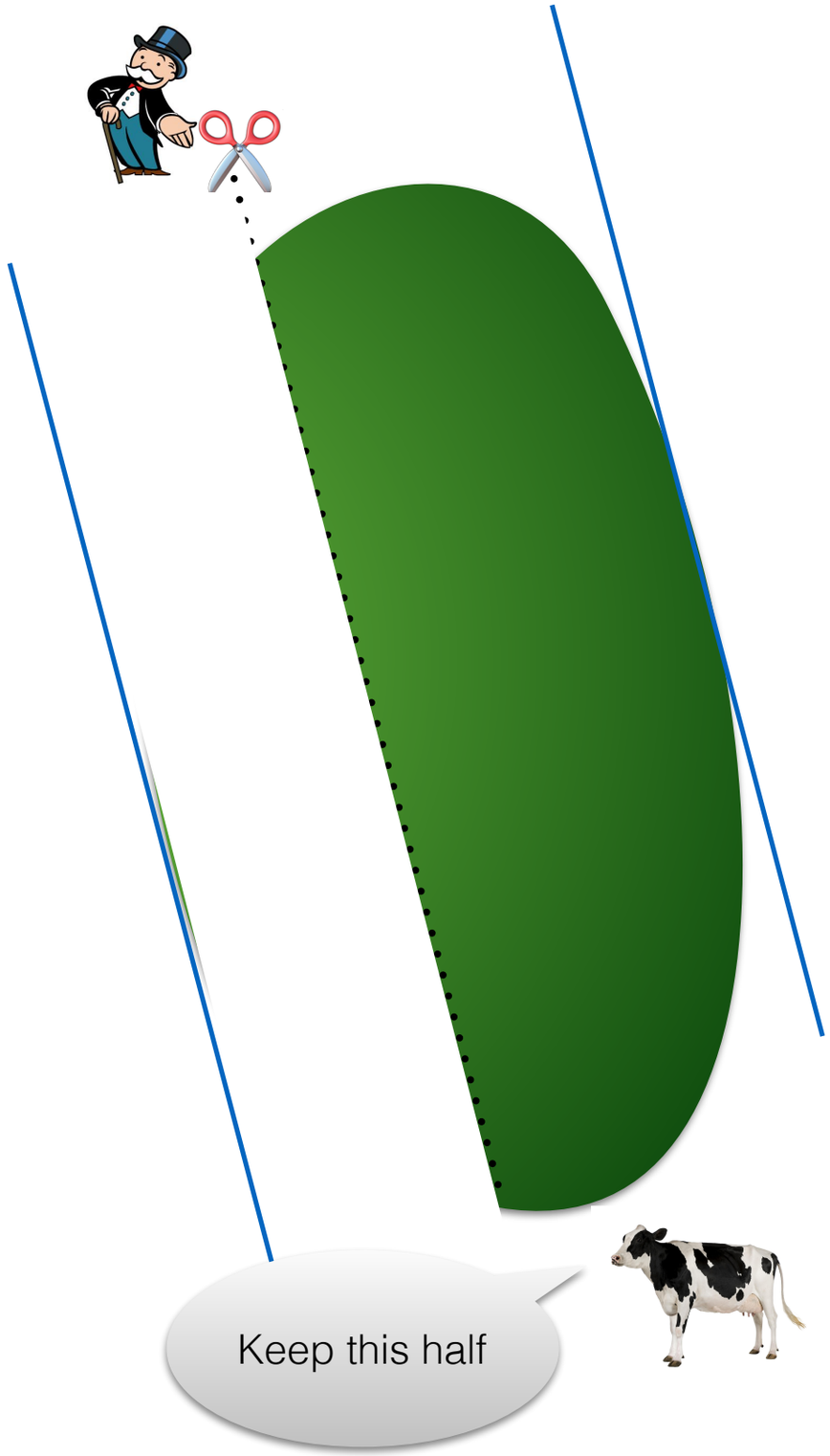
Model

- start with d-dimensional convex body (possible prices for items)
- customer gives vector of item quantities u, $\|u\| = 1$, with the promise that $w(K, u) \geq \epsilon$, asks for price
- player chooses price c, and partitions $K = K_1 \cup K_2$, where $K_1 = \{x \in K : x^\top u \leq c\}$ and $K_2 = K \setminus K_1$



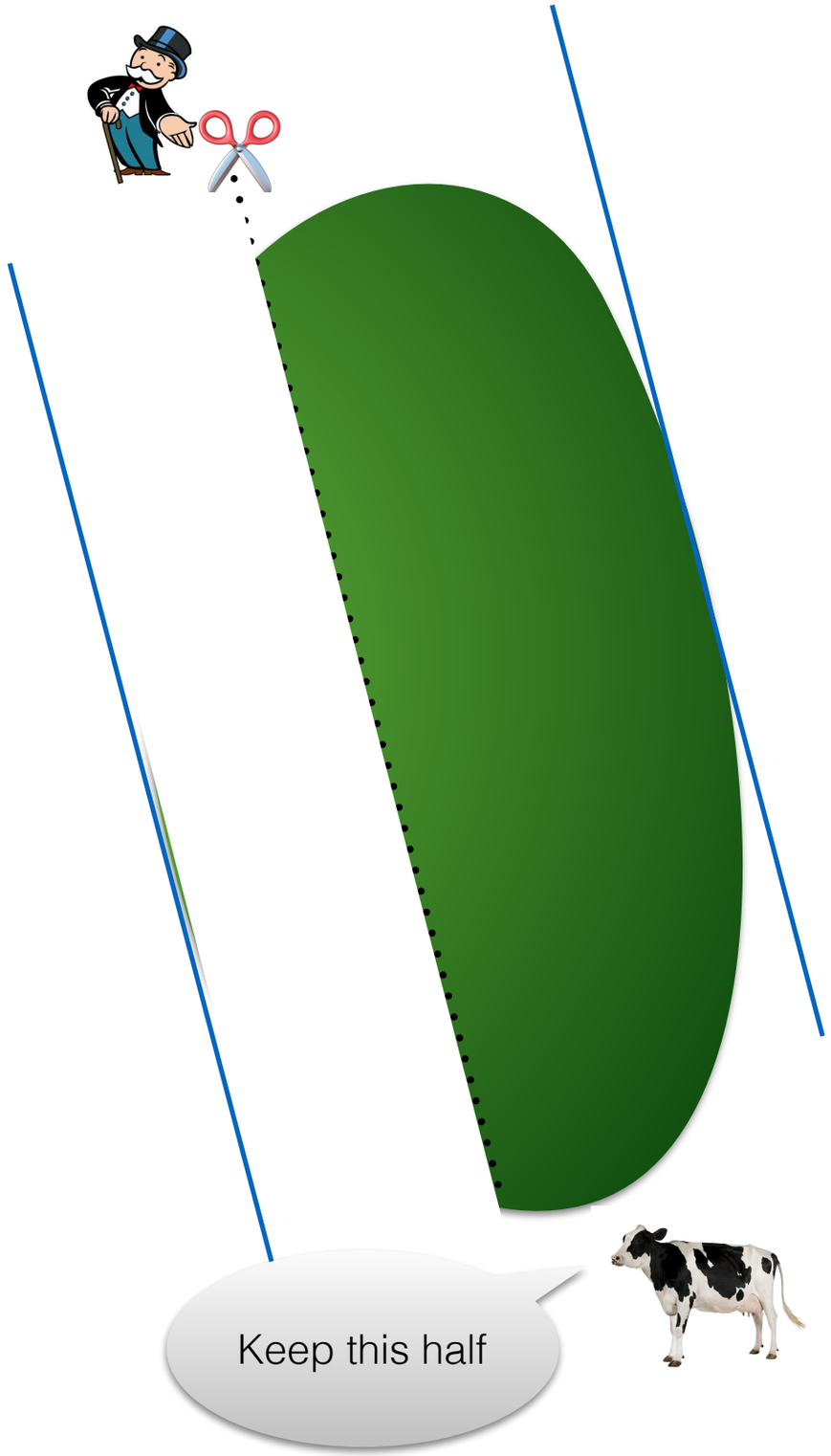
Model

- start with d-dimensional convex body (possible prices for items)
- customer gives vector of item quantities u, $\|u\| = 1$, with the promise that $w(K, u) \geq \epsilon$, asks for price
- player chooses price c, and partitions $K = K_1 \cup K_2$, where $K_1 = \{x \in K : x^\top u \leq c\}$ and $K_2 = K \setminus K_1$
- customer picks which half contains the true prices, update K accordingly



Model

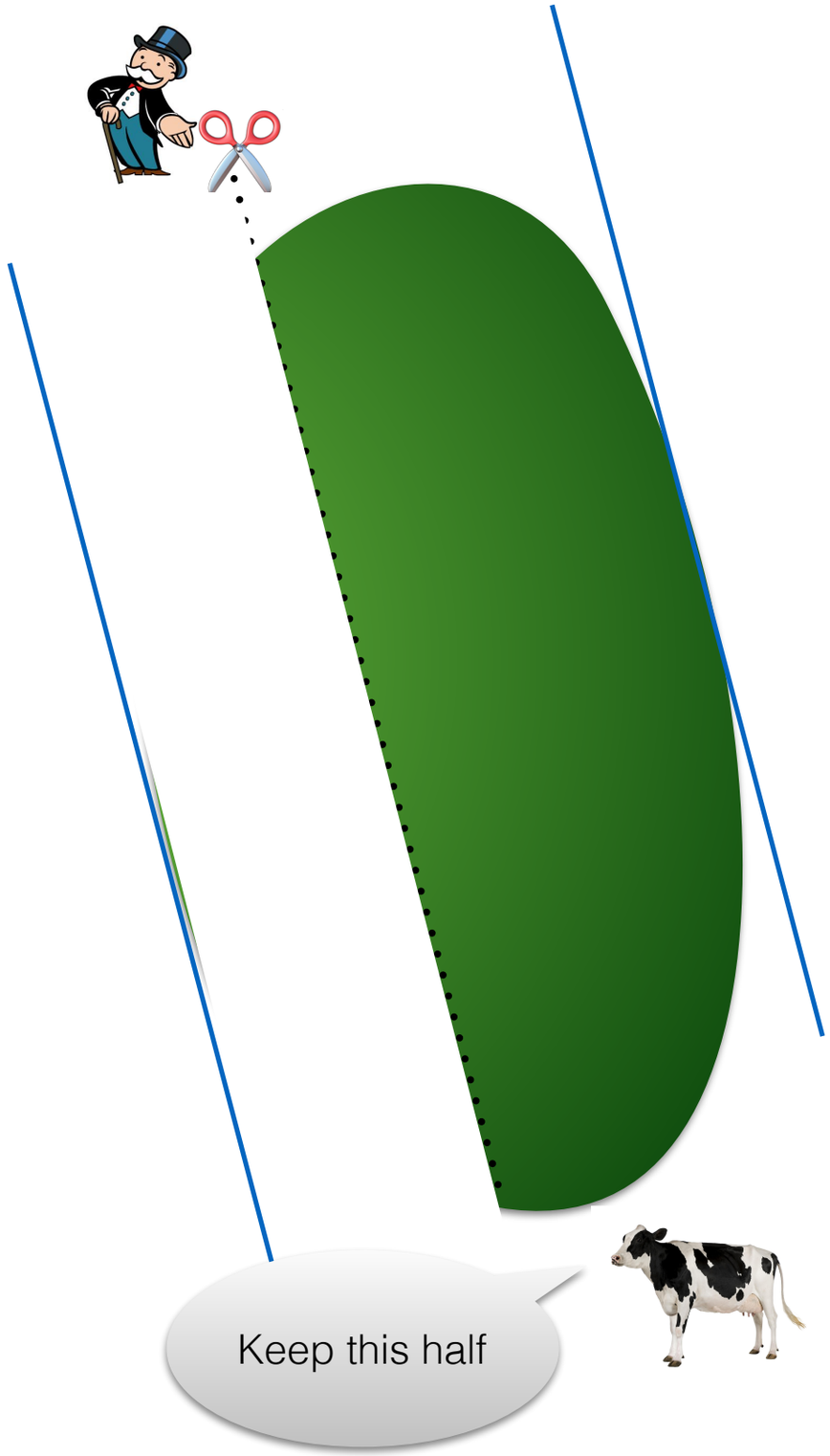
- start with d-dimensional convex body (possible prices for items)
- customer gives vector of item quantities u, $\|u\| = 1$, with the promise that $w(K, u) \geq \epsilon$, asks for price
- player chooses price c, and partitions $K = K_1 \cup K_2$, where $K_1 = \{x \in K : x^\top u \leq c\}$ and $K_2 = K \setminus K_1$
- customer picks which half contains the true prices, update K accordingly
- repeat until no valid directions remaining



Model

- start with d-dimensional convex body (possible prices for items)
- customer gives vector of item quantities u, $\|u\| = 1$, with the promise that $w(K, u) \geq \epsilon$, asks for price
- player chooses price c, and partitions $K = K_1 \cup K_2$, where $K_1 = \{x \in K : x^\top u \leq c\}$ and $K_2 = K \setminus K_1$
- customer picks which half contains the true prices, update K accordingly
- repeat until no valid directions remaining

Theorem: there exists a strategy for cutting such that **w(K,u) ≤ ε** for all **u** after **O(d log (d/ε))** iterations.



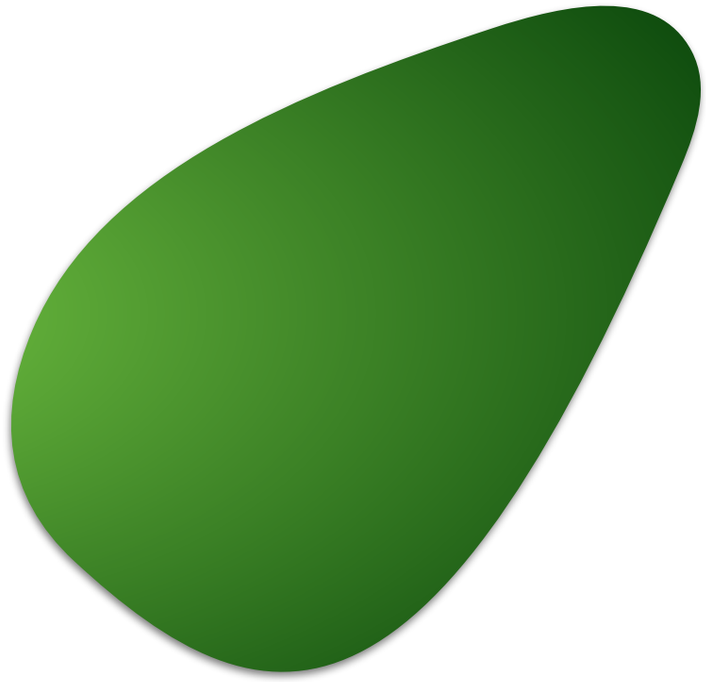
Model

- start with d-dimensional convex body (possible prices for items)
- customer gives vector of item quantities u, $\|u\| = 1$, with the promise that $w(K, u) \geq \epsilon$, asks for price
- player chooses price c, and partitions $K = K_1 \cup K_2$, where $K_1 = \{x \in K : x^\top u \leq c\}$ and $K_2 = K \setminus K_1$
- customer picks which half contains the true prices, update K accordingly
- repeat until no valid directions remaining

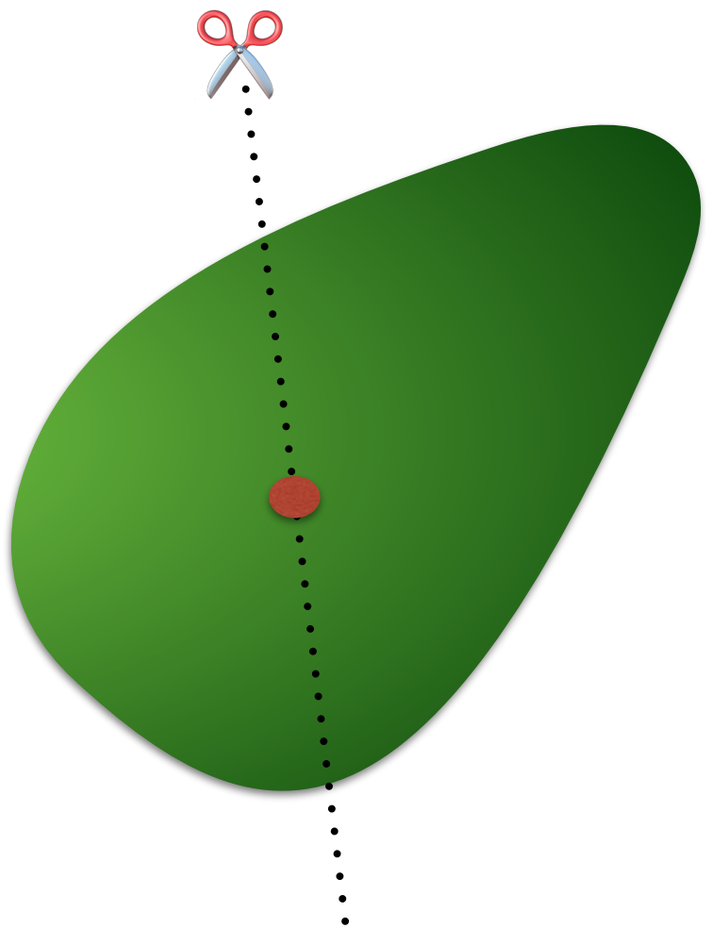
Theorem: there exists a strategy for cutting such that **w(K,u) ≤ ε** for all **u** after **O(d log (d/ε))** iterations.

Set **ε=d log T / T**, regret is at most **Tε + O(d log (d/ε)) = O(d log T)**.

Convex Geometry Toolbox

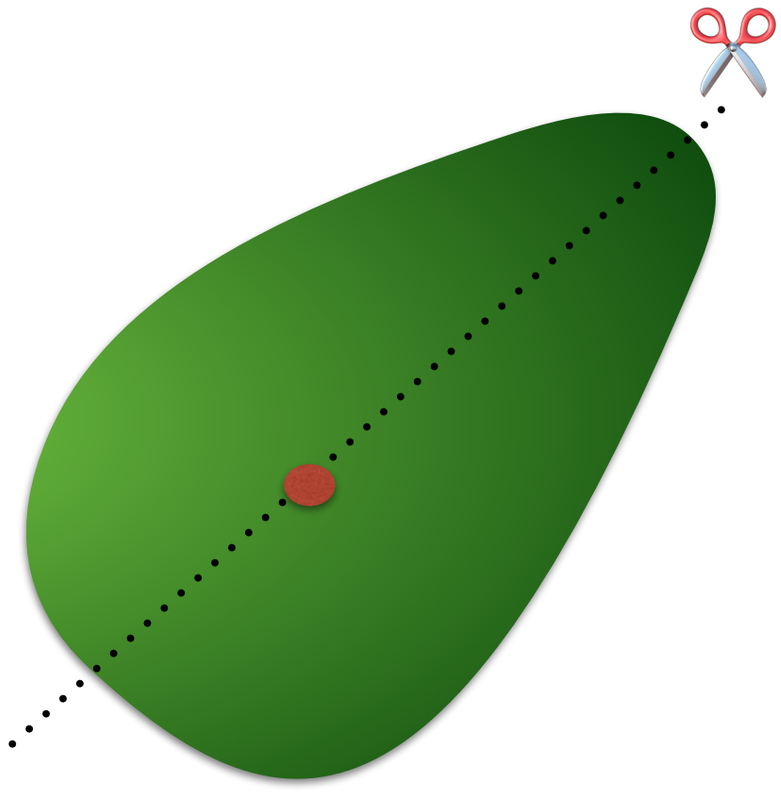


Convex Geometry Toolbox



Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

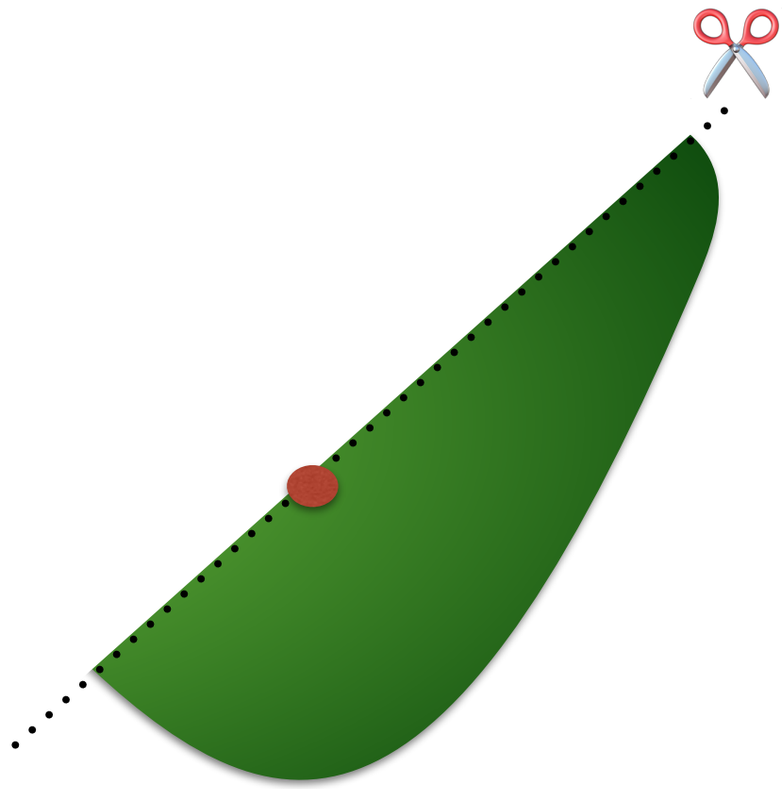
Convex Geometry Toolbox



Not quite sufficient

Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

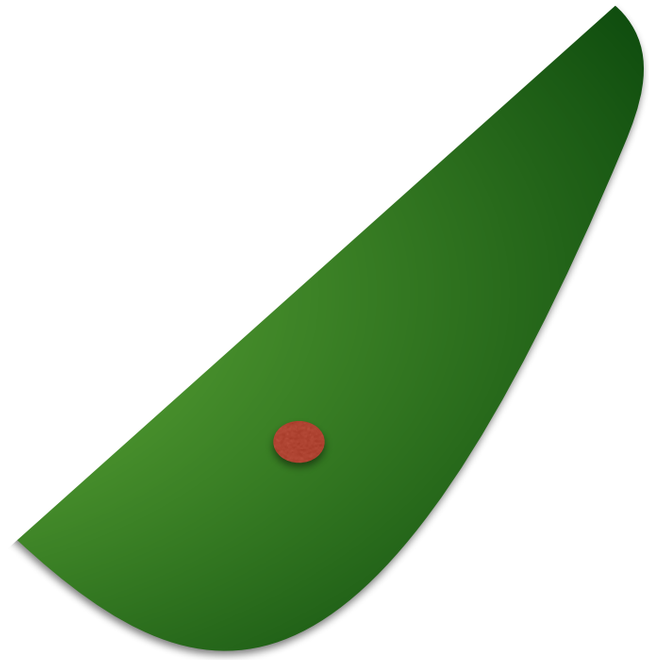
Convex Geometry Toolbox



Not quite sufficient

Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

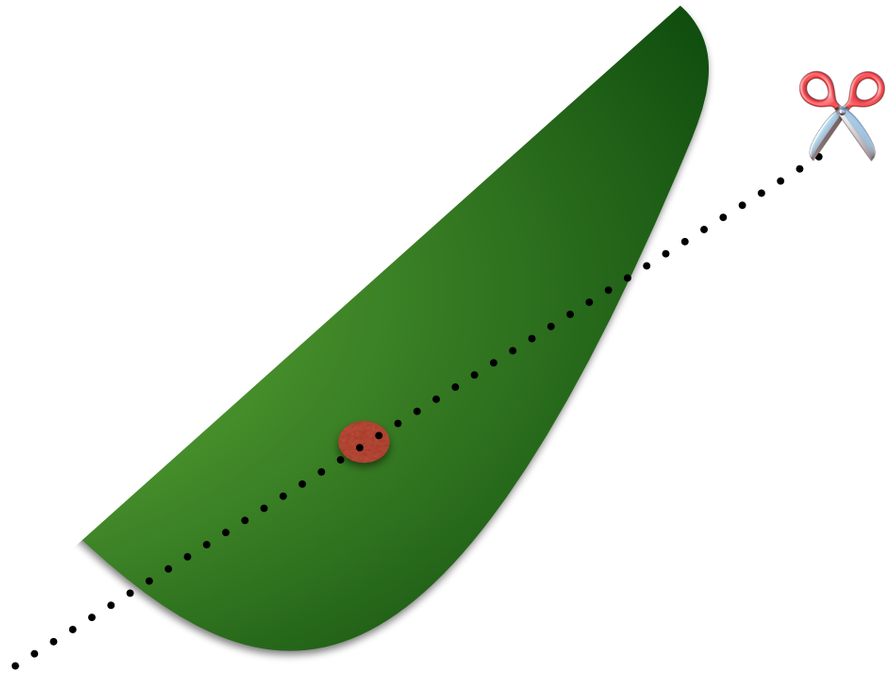
Convex Geometry Toolbox



Not quite sufficient

Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

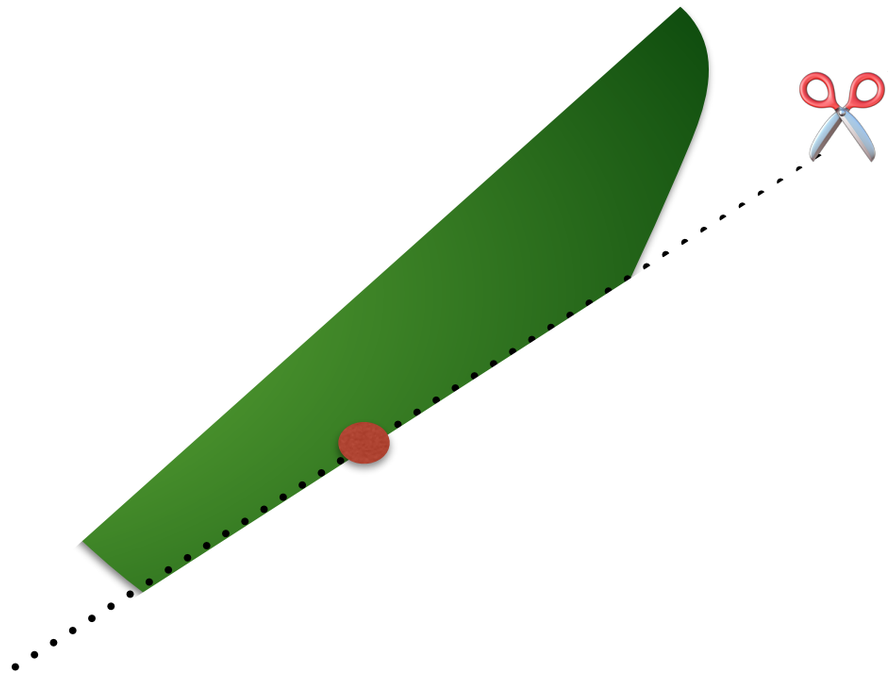
Convex Geometry Toolbox



Not quite sufficient

Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

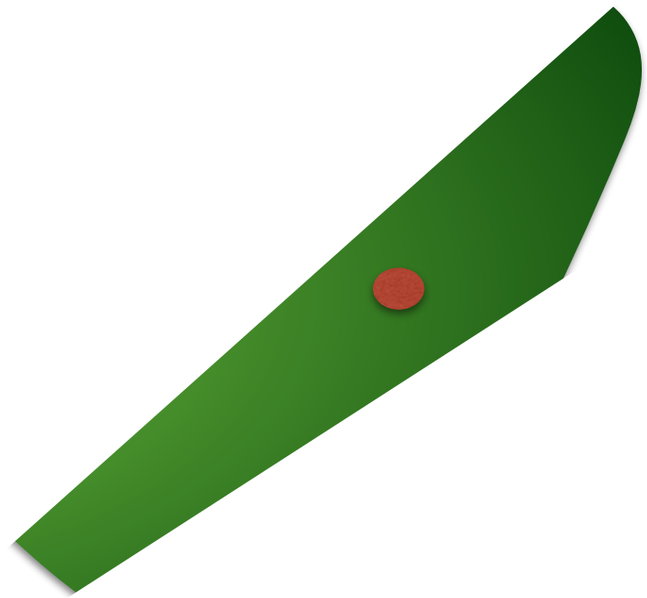
Convex Geometry Toolbox



Not quite sufficient

Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

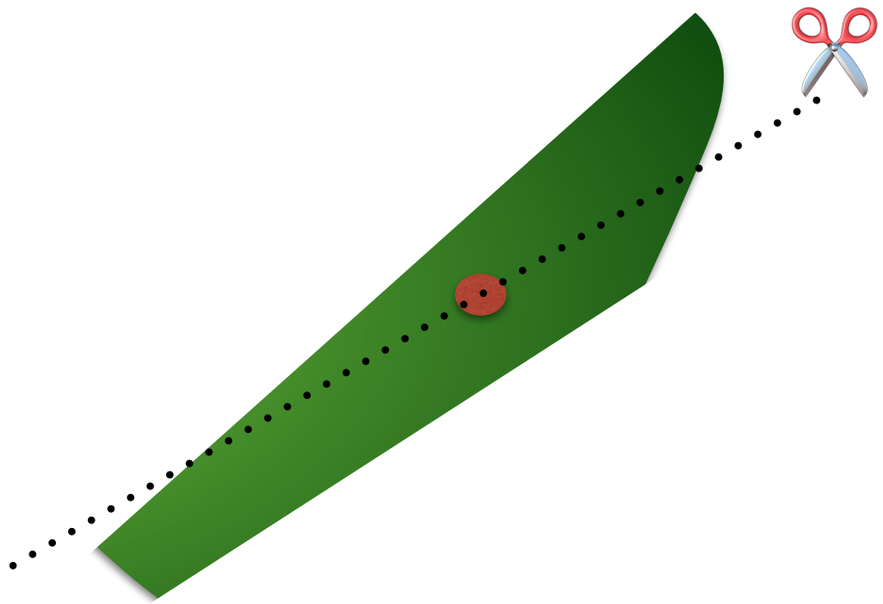
Convex Geometry Toolbox



Not quite sufficient

Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

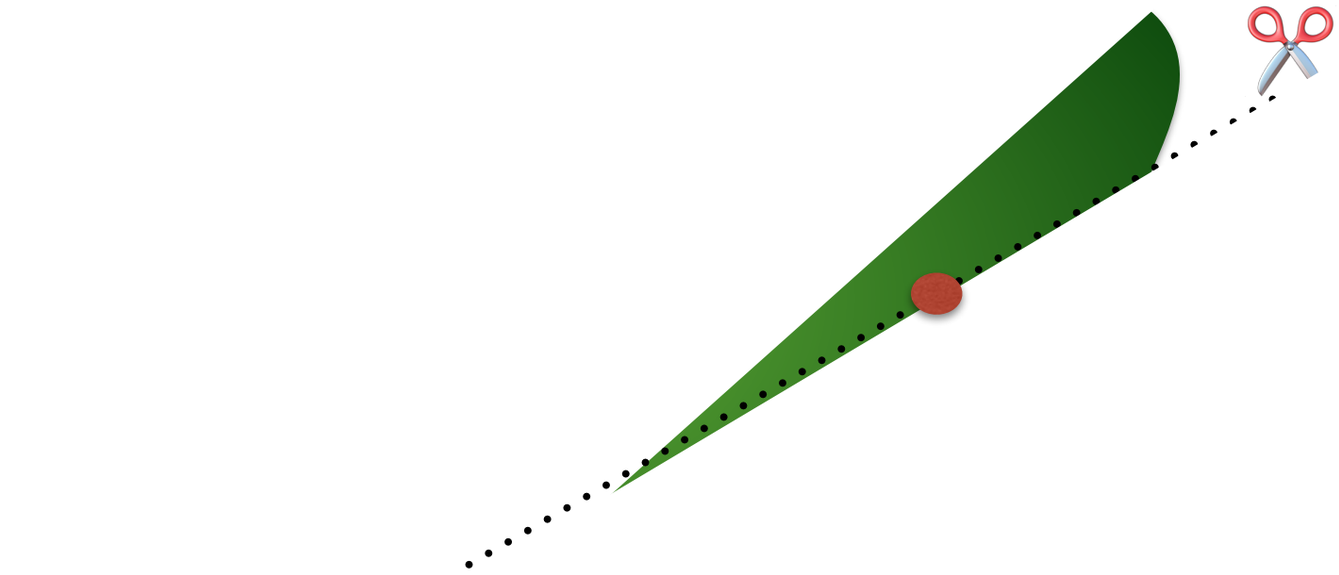
Convex Geometry Toolbox



Not quite sufficient

Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

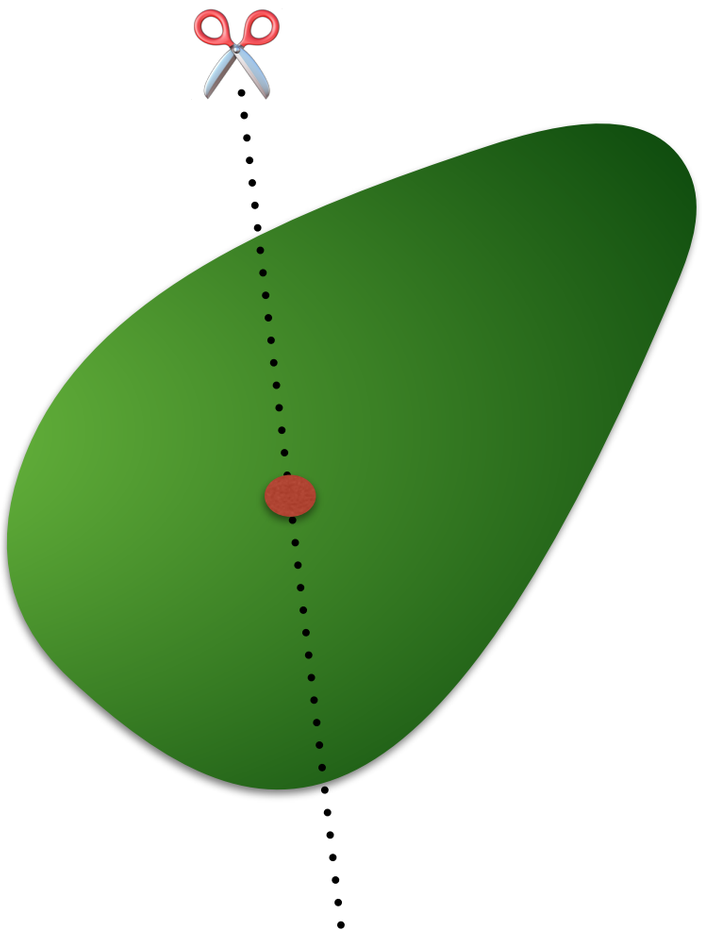
Convex Geometry Toolbox



Not quite sufficient

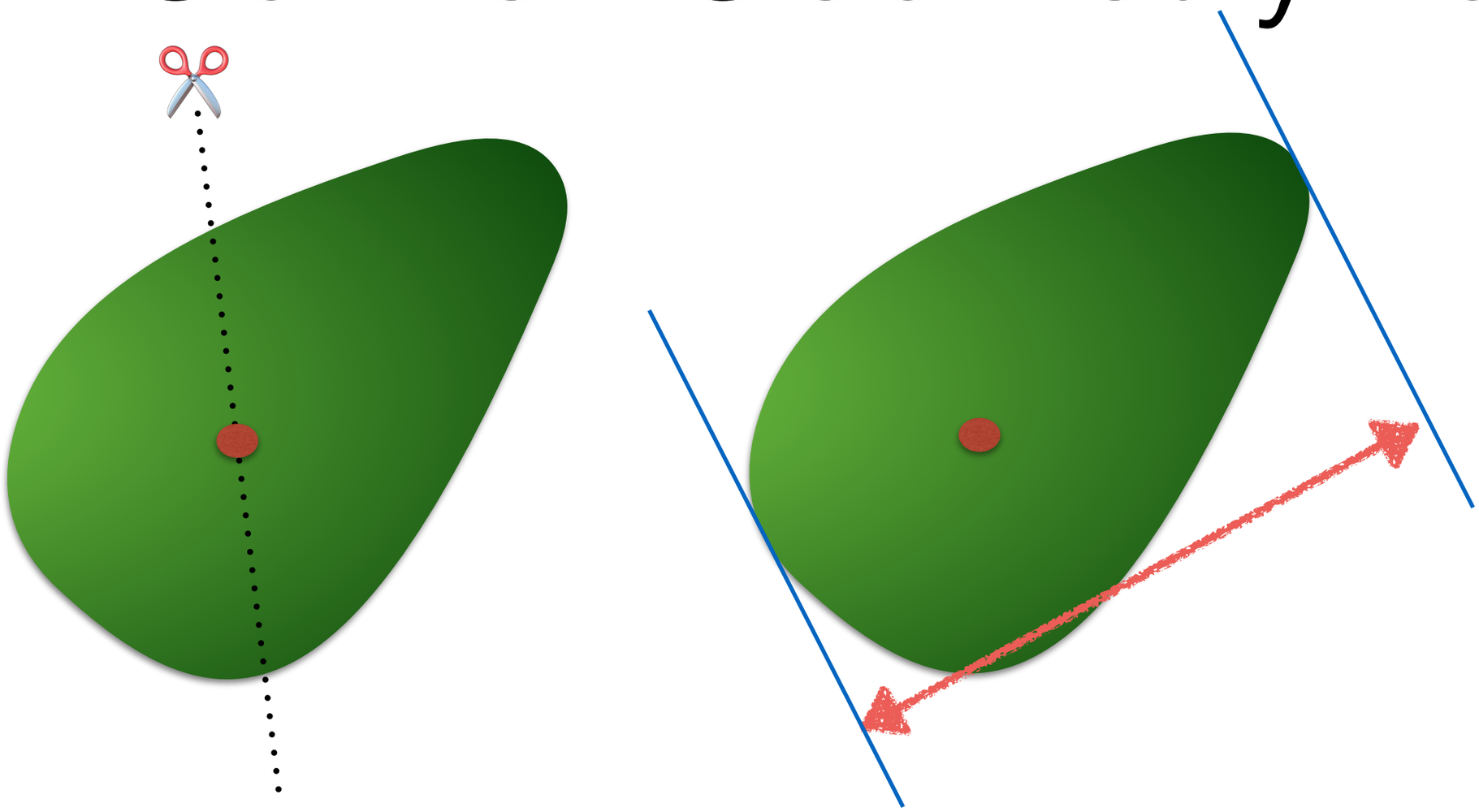
Grunbaum's Theorem: any cut through the centroid partitions **K** in two pieces of roughly equal volume (largest/smallest **≤ e**).

Convex Geometry Toolbox



Grunbaum's Theorem: Any cut through **c(K)** partitions **K** in two pieces with volume ratio at most **e**.

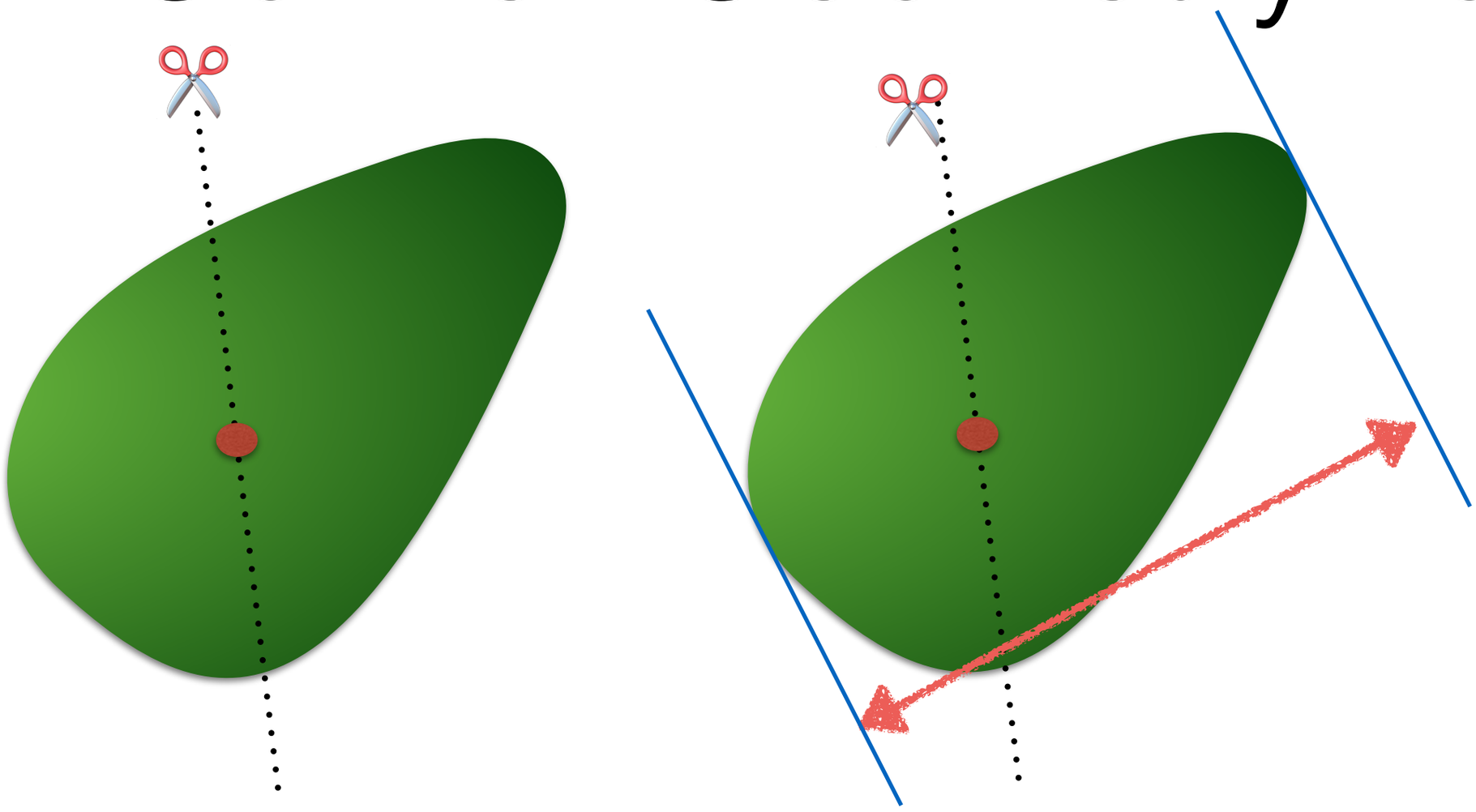
Convex Geometry Toolbox



Grunbaum's Theorem: Any cut through **c(K)** partitions **K** in two pieces with volume ratio at most **e**.

Directional Grunbaum (this work): A cut through **c(K)** partitions $\mathbf{K=K_1 \cup K_2}$ such that $\mathbf{w(K_i) \geq w(K,u) / (d+1)}$, for all **u**.

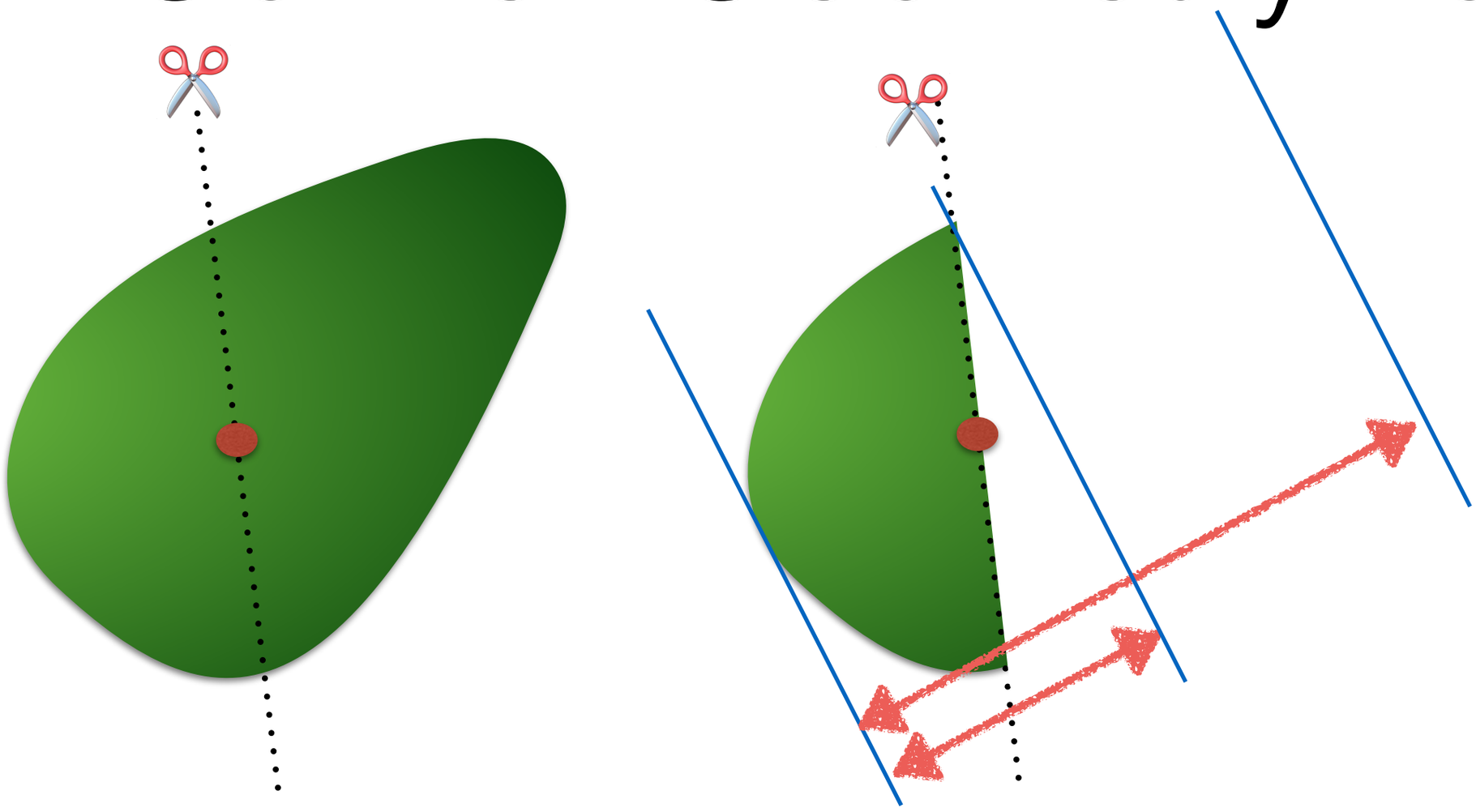
Convex Geometry Toolbox



Grunbaum's Theorem: Any cut through **c(K)** partitions **K** in two pieces with volume ratio at most **e**.

Directional Grunbaum (this work): A cut through **c(K)** partitions $\mathbf{K=K_1 \cup K_2}$ such that $\mathbf{w(K_i) \geq w(K,u) / (d+1)}$, for all **u**.

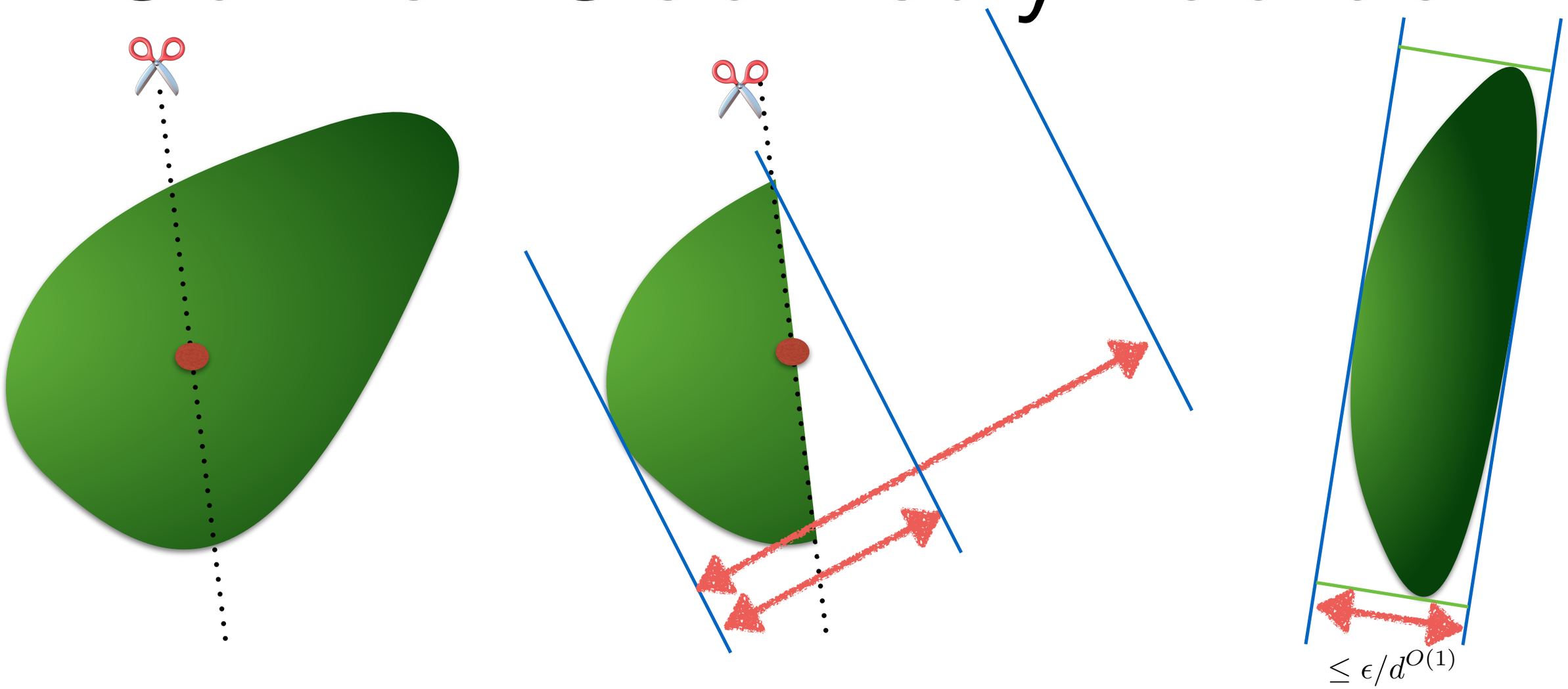
Convex Geometry Toolbox



Grunbaum's Theorem: Any cut through **c(K)** partitions **K** in two pieces with volume ratio at most **e**.

Directional Grunbaum (this work): A cut through **c(K)** partitions $\mathbf{K=K_1 \cup K_2}$ such that $\mathbf{w(K_i) \geq w(K,u) / (d+1)}$, for all **u**.

Convex Geometry Toolbox



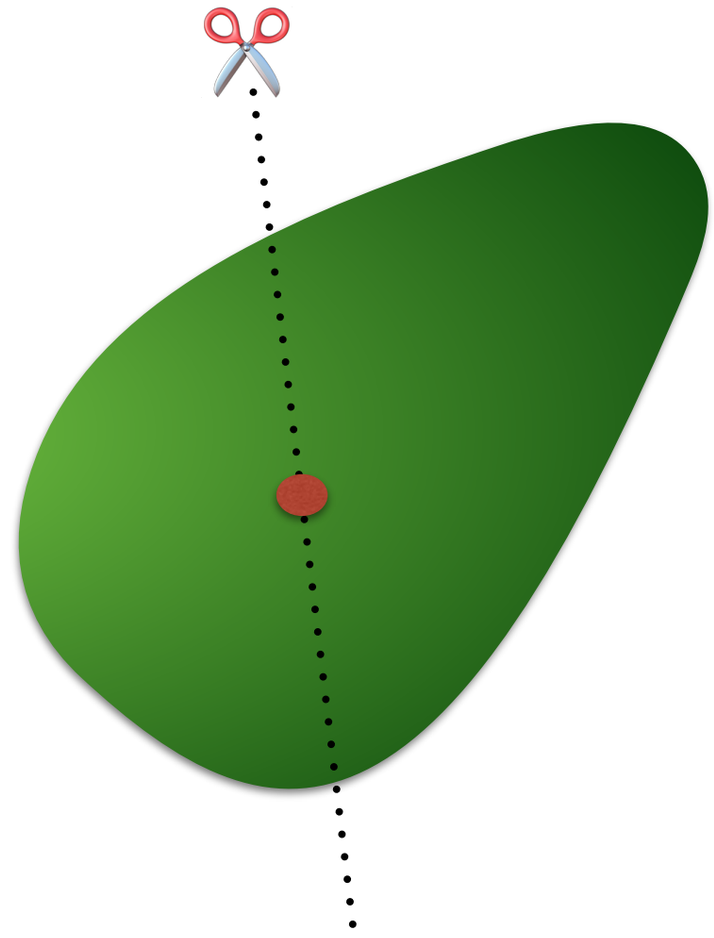
Grunbaum's Theorem: Any cut through **c(K)** partitions **K** in two pieces with volume ratio at most **e**.

Directional Grunbaum **(this work)**: A cut through **c(K)** partitions **$K=K_1 \cup K_2$** such that **$w(K_i) \geq w(K,u) / (d+1)$**, for all **u**.

Cylindrification **(this work)**: If **$w(K,u) \geq \delta$** for all **u**, and **S** is **(d-1)**-dimensional, then **$\mathrm{Vol}(\mathrm{Proj}_S K) \leq (d+1)^2/\delta \, \mathrm{Vol}(K)$**.

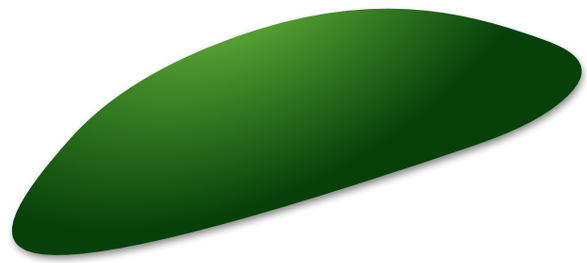
Algorithm

1. Cut through centroid.



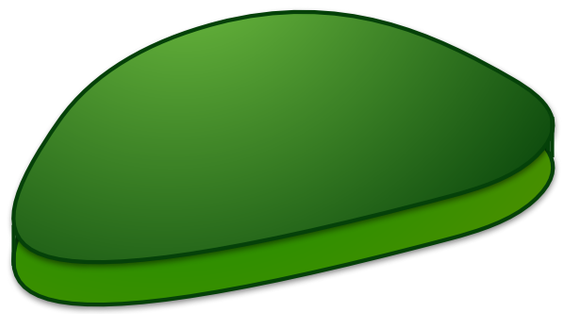
Algorithm

1. Cut through centroid.
2. When $w(K, u) \leq \delta = \epsilon/d^{O(1)}$, cylindrify.



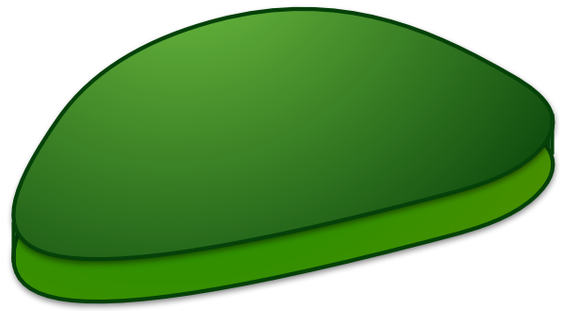
Algorithm

1. Cut through centroid.
2. When $w(K, u) \leq \delta = \epsilon / d^{O(1)}$, cylindrify.



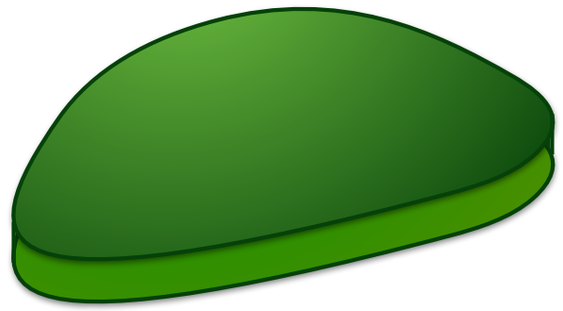
Algorithm

1. Cut through centroid.
2. When $w(K,u) \leq \delta = \epsilon/d^{O(1)}$, cylindrify.
3. Small volume blow-up from $\leq d$ cylindrifications, overall need to reduce volume by a factor of $(d/\epsilon)^{O(d)}$.



Algorithm

1. Cut through centroid.
2. When $w(K, u) \leq \delta = \epsilon/d^{O(1)}$, cylindrify.
3. Small volume blow-up from $\leq d$ cylindrifications, overall need to reduce volume by a factor of $(d/\epsilon)^{O(d)}$.

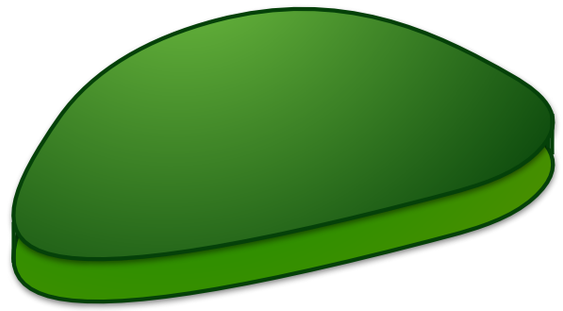


POLYNOMIAL TIME ALGORITHM

1. Compute approximate centroid, requires almost uniform sampling [**LS '93**]

Algorithm

1. Cut through centroid.
2. When $w(K, u) \leq \delta = \epsilon/d^{O(1)}$, cylindrify.
3. Small volume blow-up from $\leq d$ cylindrifications, overall need to reduce volume by a factor of $(d/\epsilon)^{O(d)}$.

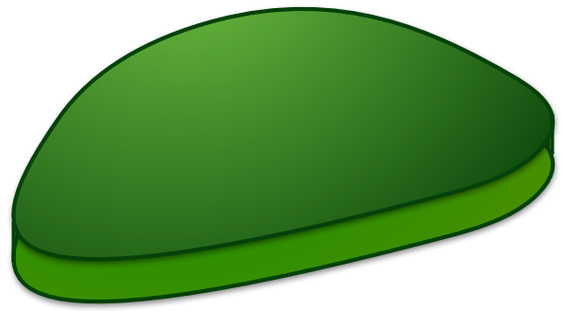


POLYNOMIAL TIME ALGORITHM

1. Compute approximate centroid, requires almost uniform sampling [**LS '93**]
2. Robust versions of our theorems

Algorithm

1. Cut through centroid.
2. When $w(K,u) \leq \delta = \epsilon/d^{O(1)}$, cylindrify.
3. Small volume blow-up from $\leq d$ cylindrifications, overall need to reduce volume by a factor of $(d/\epsilon)^{O(d)}$.



POLYNOMIAL TIME ALGORITHM

1. Compute approximate centroid, requires almost uniform sampling [**LS '93**]
2. Robust versions of our theorems

Open: obtain regret **O(d log d + d log log T)** by combining with the 1-dimensional trick.

Thank you!