Learning What Works Best When

Swati Gupta
Simons Institute, UC Berkeley
Georgia Institute of Technology

Joint work with Michel Goemans (MIT), Patrick Jaillet (MIT),
Iain Dunning (Deepmind), John Silberholz (Ross School of Business)

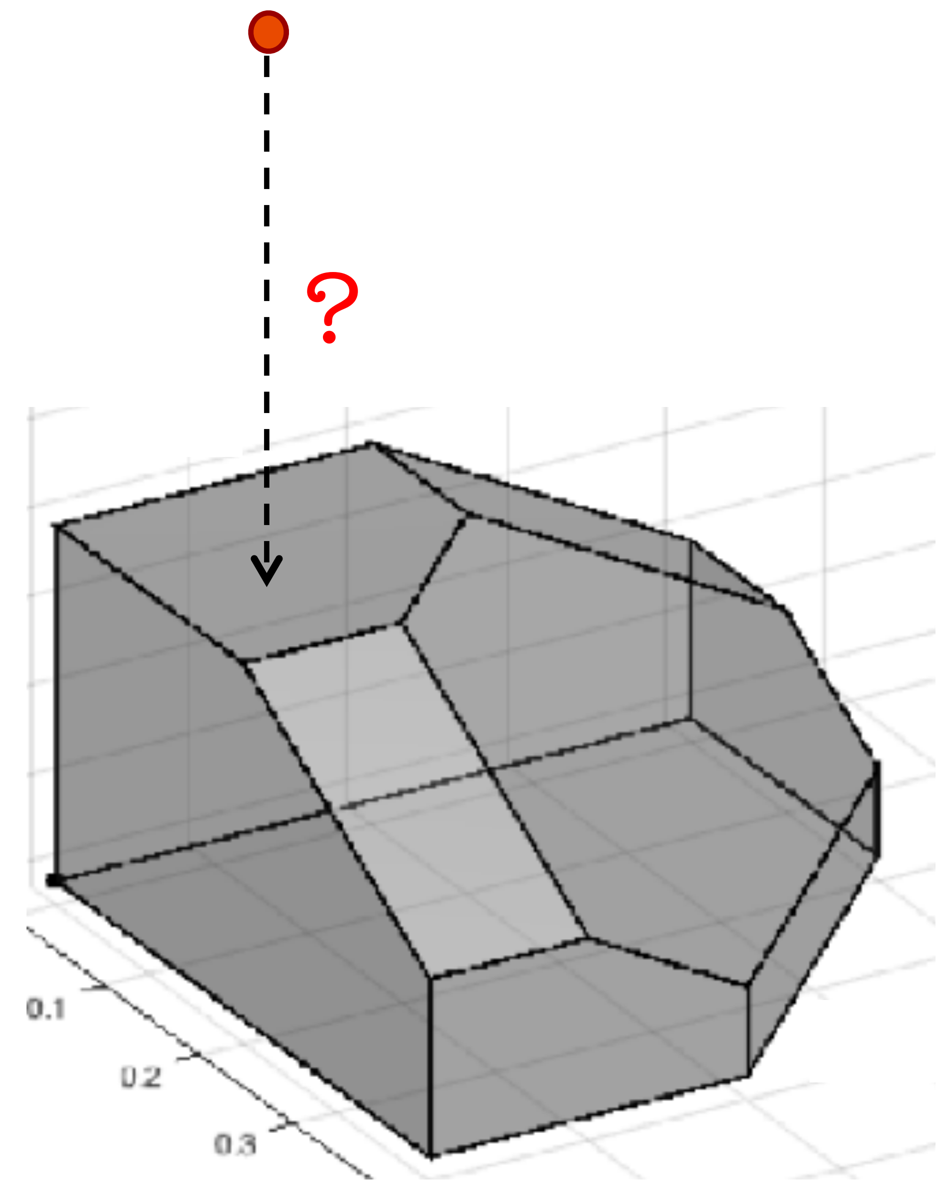
05 + 02 + 2=01+8

Mathematical and Computational Challenges in Real-Time Decision Making

Three fundamental questions

Three fundamental questions

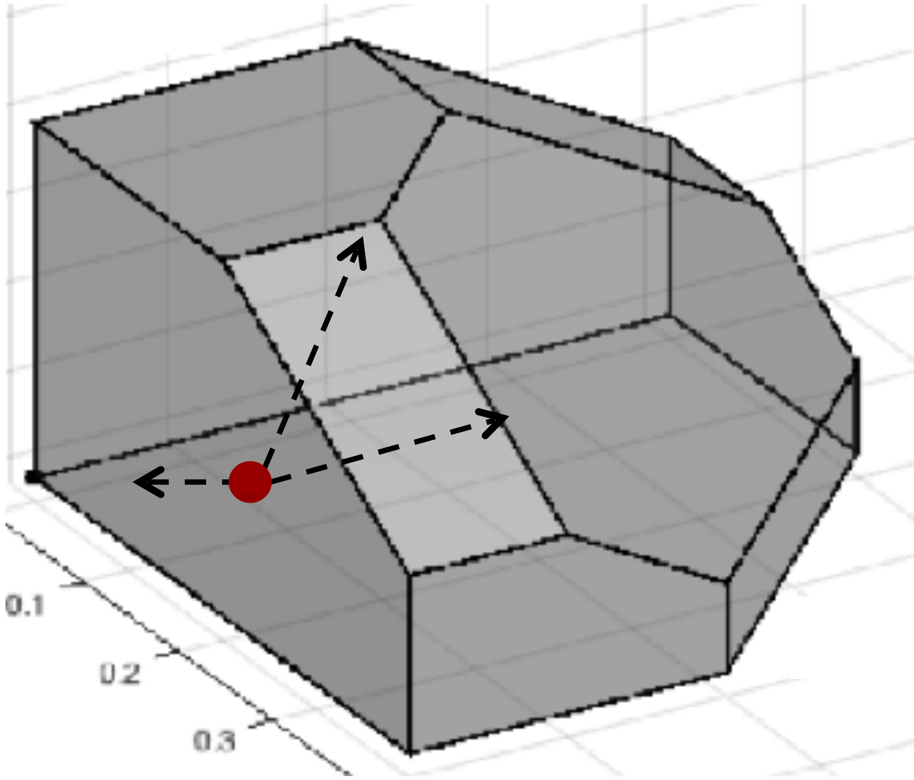
(a) How to compute projections?



Three fundamental questions

(a) How to compute projections?

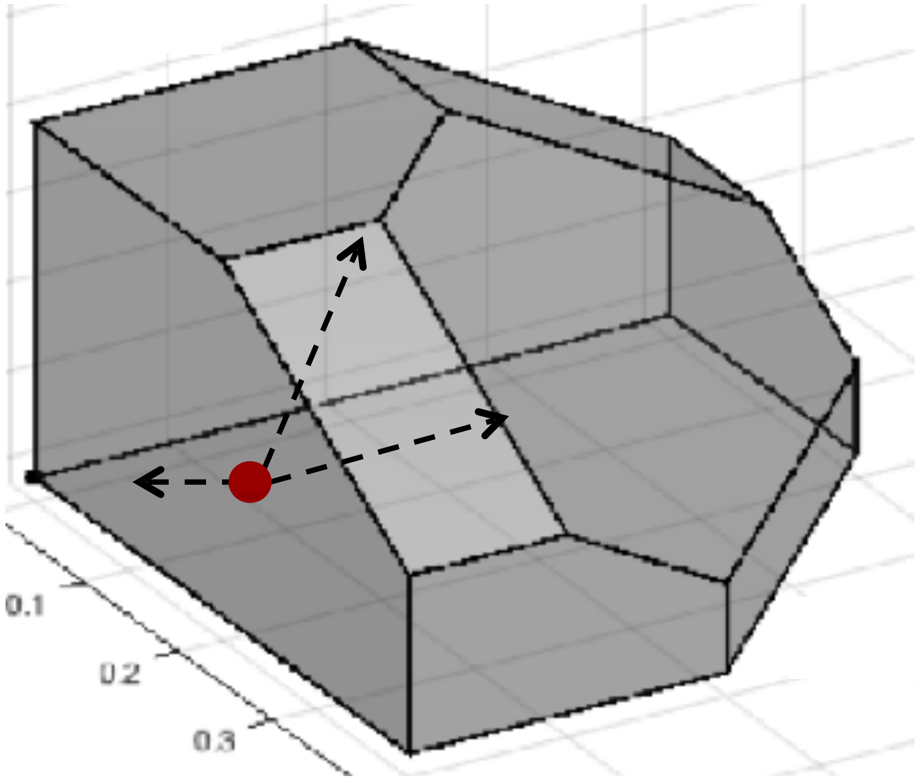
(b) How far can we move along a direction while staying feasible?



Three fundamental questions

(a) How to compute projections?

(b) How far can we move along a direction while staying feasible?

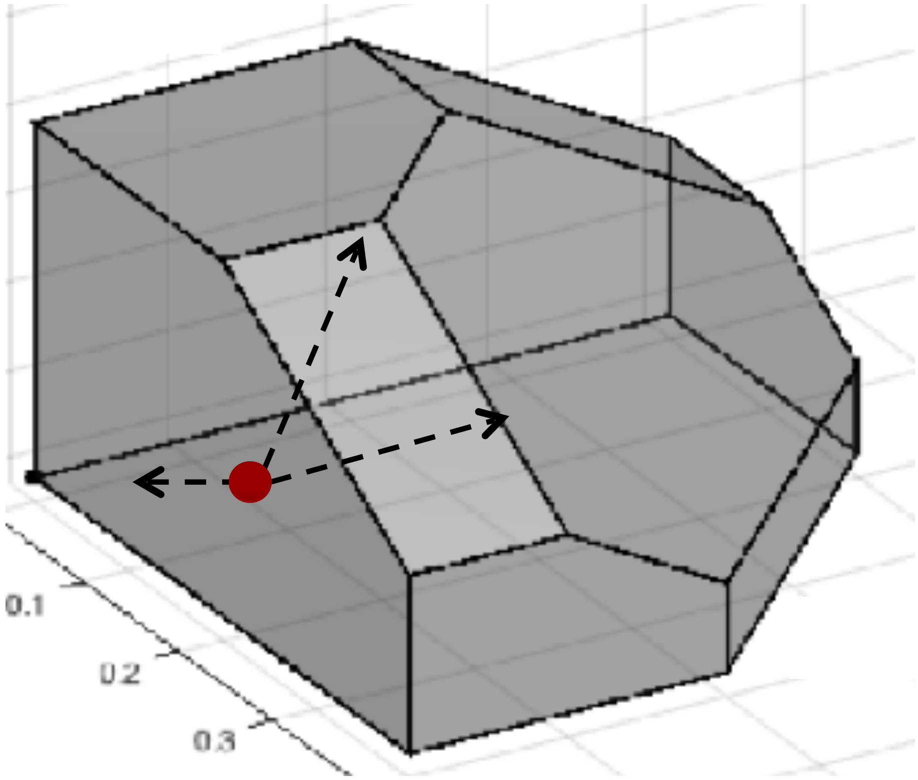


(c) Can we learn which algorithm works best on an unseen instance?

Three fundamental questions

(a) How to compute **projections?**

(b) How far can we move along a direction while staying feasible?



(c) Can we learn which algorithm works best on an unseen instance?

Why are projections important?

Key step in many algorithms across

- Online Learning

  Game Theory

  Machine Learning

  Stochastic optimization

  Robust optimization

  …

→
- Problem setup
- Examples
- Online Mirror Descent
  - Projection!

Online Learning

Online Learning

Historic Data
Or Prior knowledge

Repeats over time

Select an action
Or a decision

Incur losses or gains in a **dynamic environment**

Online Learning

Historic Data
Or Prior knowledge

Repeats over time

Select an action
Or a decision

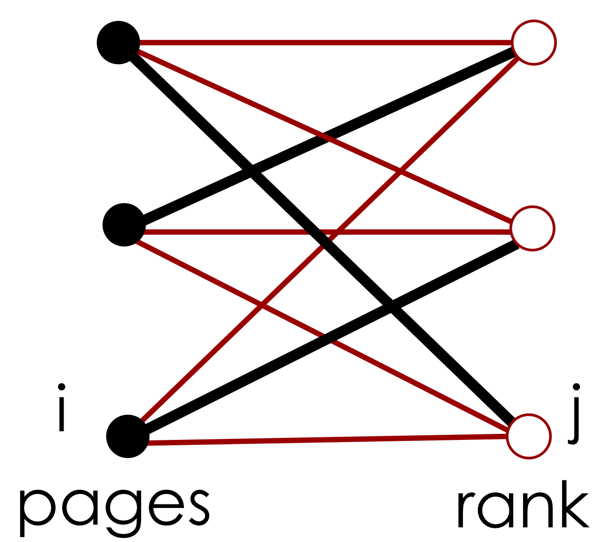
Incur losses or gains in a
dynamic environment

How to perform well compared to best fixed decision in hindsight?

Online Learning

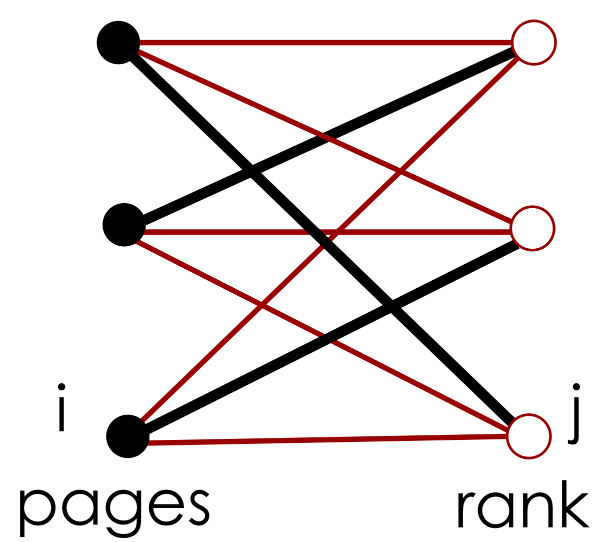
Online Learning

Matchings



Online Learning

Matchings

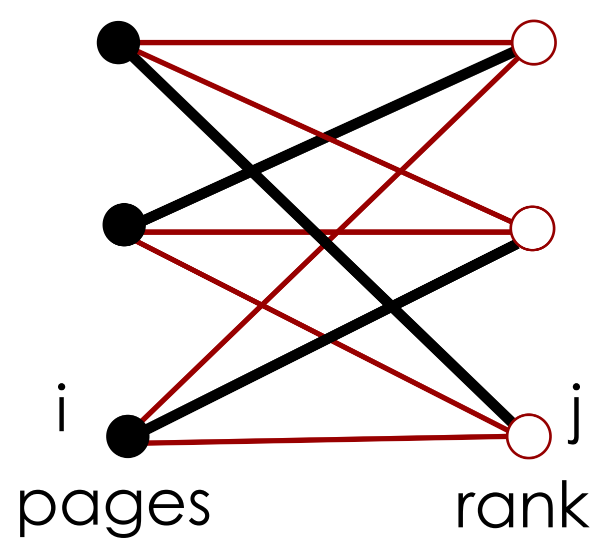


Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

Online Learning

Matchings

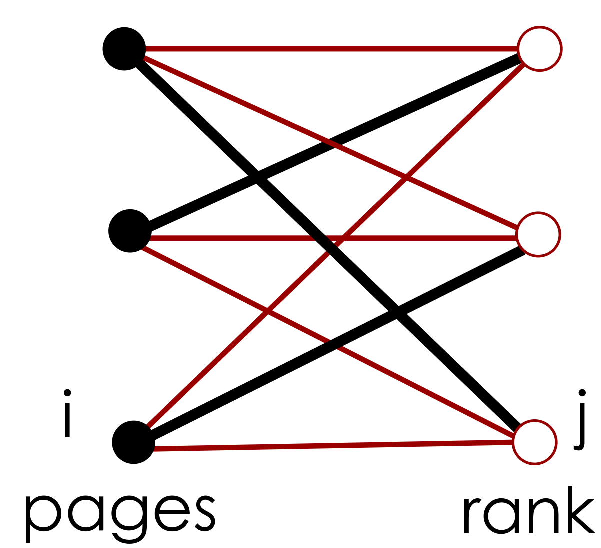


Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

Online Learning

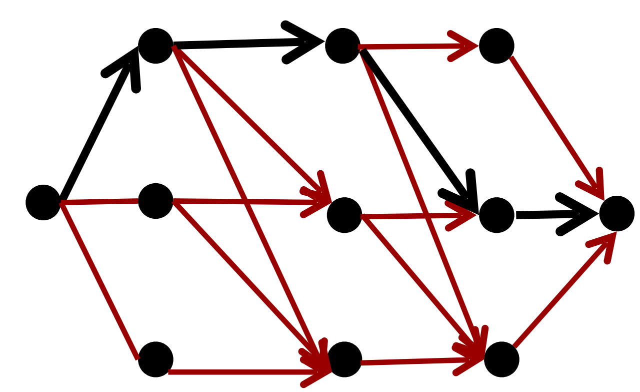
Matchings



Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

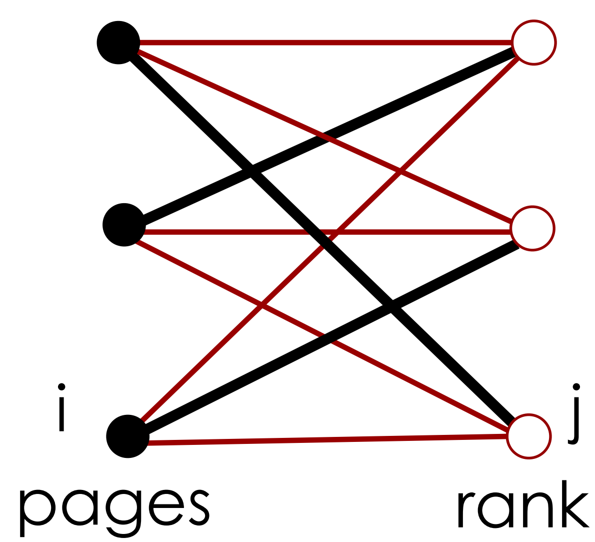
s-t paths



[Cohen, **Gupta**, Kalas, Perakis, '16]

Online Learning

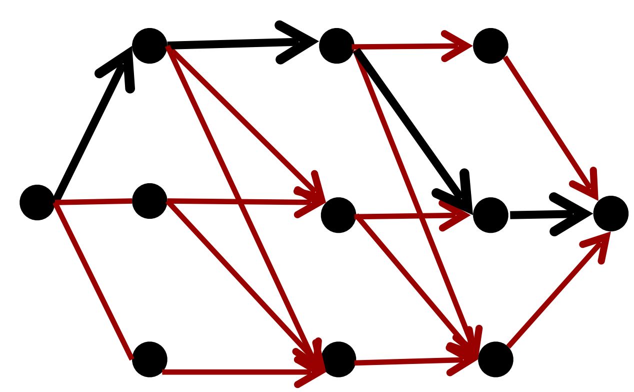
Matchings



Permutations

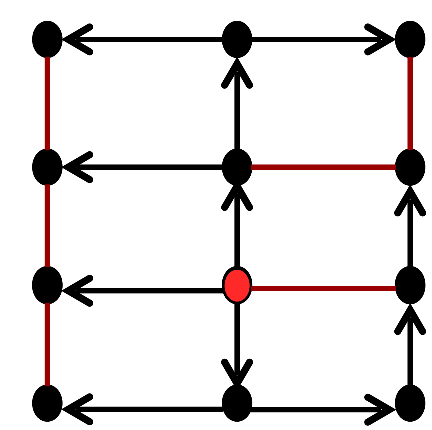
1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

s-t paths



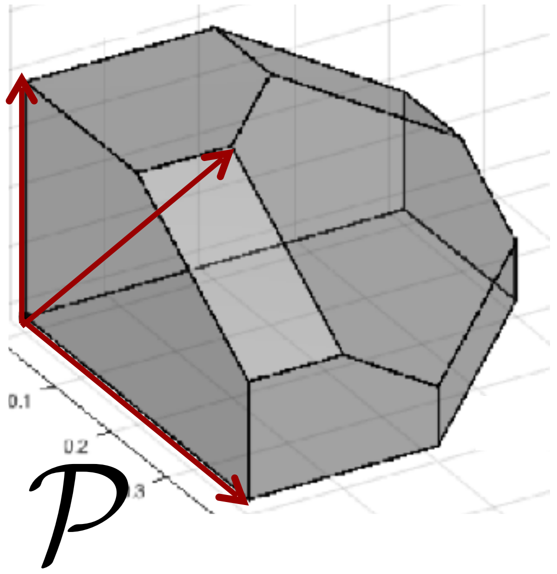
[Cohen, **Gupta**, Kalas, Perakis, ‘16]

Spanning Trees



Online Learning

Decision Space



allow convex combinations, sample at random

Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

Online Learning

Online Learning Framework

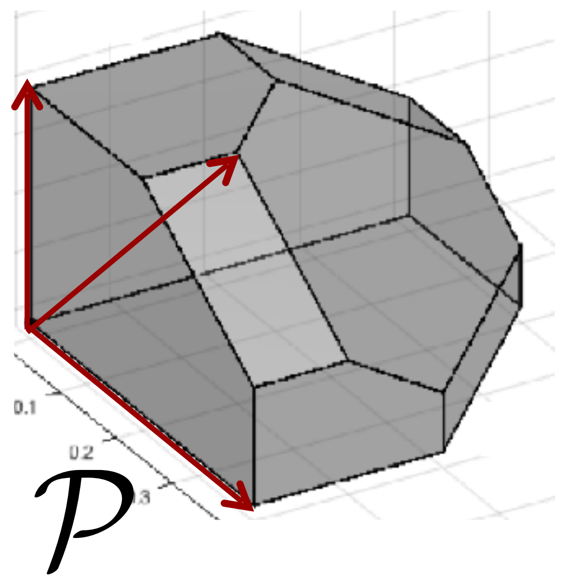
- learner chooses a decision,

  $x_t \in \mathcal{P}$
- a linear loss revealed,

  $l_t \in \mathcal{L} : \mathcal{P} \to \mathbb{R}$
- the loss incurred for time **t**:

  $l_t(x_t)$

Decision Space



allow convex combinations, sample at random

Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

Example

Online Learning Framework

- learner chooses a decision,

$$x_t \in \mathcal{P}$$

- a linear loss revealed,

$$l_t \in \mathcal{L} : \mathcal{P} \to \mathbb{R}$$

- the loss incurred for time **t**:

$$l_t(x_t)$$

Suppose
x_t = (2, 3 ,1, 4)

↓

Page 1 at rank 2
Page 2 at rank 3
Page 3 at rank 1
Page 4 at rank 4

↓

Display:

Page 3
Page 1
Page 2
Page 4

Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

Example

Online Learning Framework

- learner chooses a decision,

$$x_t \in \mathcal{P}$$

- a linear loss revealed,

$$l_t \in \mathcal{L} : \mathcal{P} \to \mathbb{R}$$

- the loss incurred for time **t**:

$$l_t(x_t)$$

Suppose
$\mathbf{x_t}$ = (2, 3 ,1, 4)

↓

Page 1 at rank 2
Page 2 at rank 3
Page 3 at rank 1
Page 4 at rank 4

↓

Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

Display:

Observe user clicks:

Page 3 <----> 20%
Page 1 <----> 40%
Page 2 <----> 30%
Page 4 <----> 10%

$$p_t = \begin{bmatrix} 0.40 \\ 0.30 \\ 0.20 \\ 0.10 \end{bmatrix}$$

Example

Online Learning Framework

- learner chooses a decision,

  $$x_t \in \mathcal{P}$$

- a linear loss revealed,

  $$l_t \in \mathcal{L} : \mathcal{P} \to \mathbb{R}$$

- the loss incurred for time **t**:

  $$l_t(x_t)$$

Example Loss Function:

$$l_t(x) = p_t \cdot x$$

Penalizes if a highly desired page is put later in the ranking

Suppose
x_t = (2, 3 ,1, 4)

↓

Page 1 at rank 2
Page 2 at rank 3
Page 3 at rank 1
Page 4 at rank 4

Permutations

1, 2, 3, 4,
2, 3, 1, 4,
3, 1, 4, 2,
4, 1, 3, 2,
2, 3, 4, 1…

↓

Display: **Observe user clicks:**

Page 3 <----> 20%
Page 1 <----> 40%
Page 2 <----> 30%
Page 4 <----> 10%

$$p_t = \begin{bmatrix} 0.40 \\ 0.30 \\ 0.20 \\ 0.10 \end{bmatrix}$$

Loss for x_t
= **2*0.40** (page 1) + **3*0.30** (page 2) + **1*0.20** (page 3) **+ 4*0.10** (page 4).

Bottleneck in First-Order **Projection**-Based Algorithms

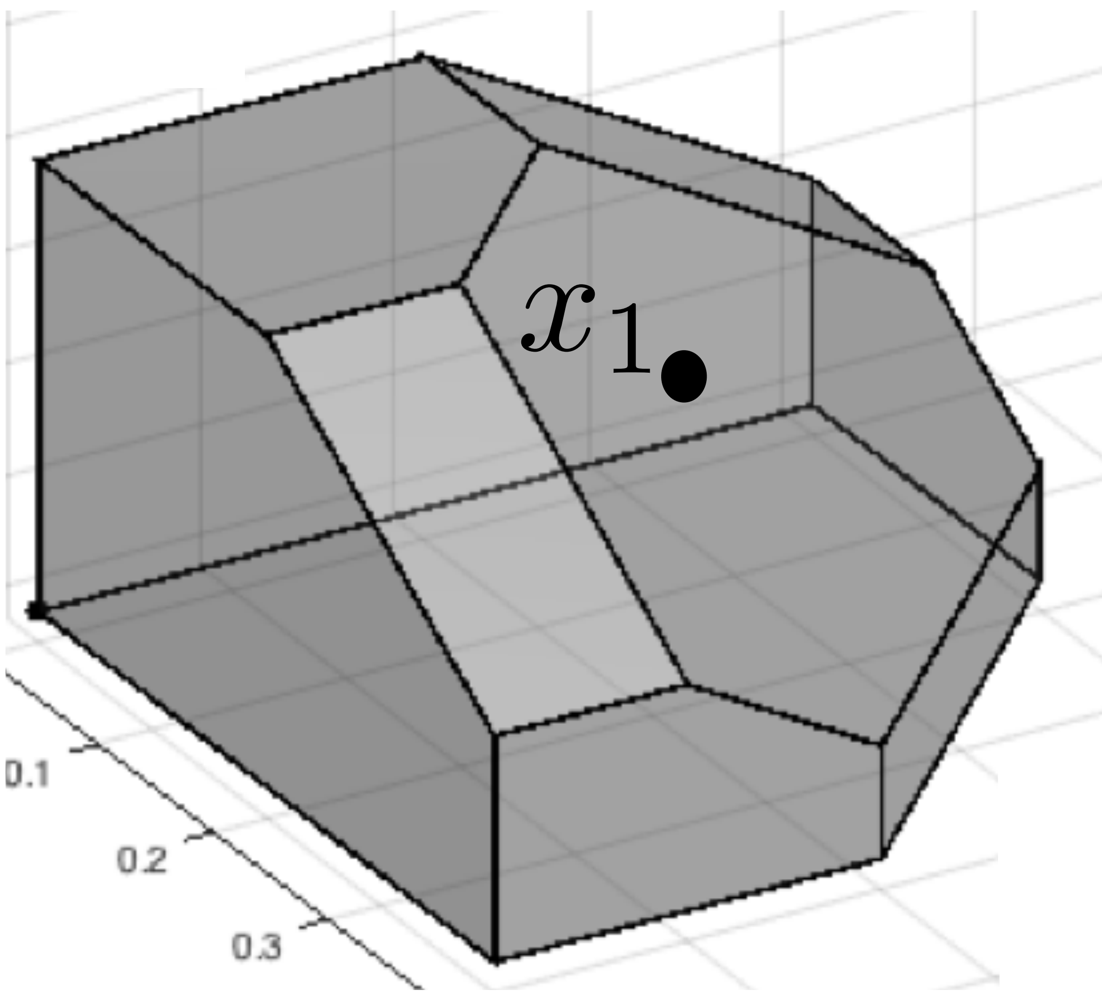
Online Mirror Descent

[Zinkevich 2003], [Nemirovski, Yudin 1983]

Optimal regret in many cases
[for e.g. Srebro, Sridharan, Tewari 2010]

*But **Computationally Slow**!*

Bottleneck in First-Order **Projection**-Based Algorithms



constrained decision set

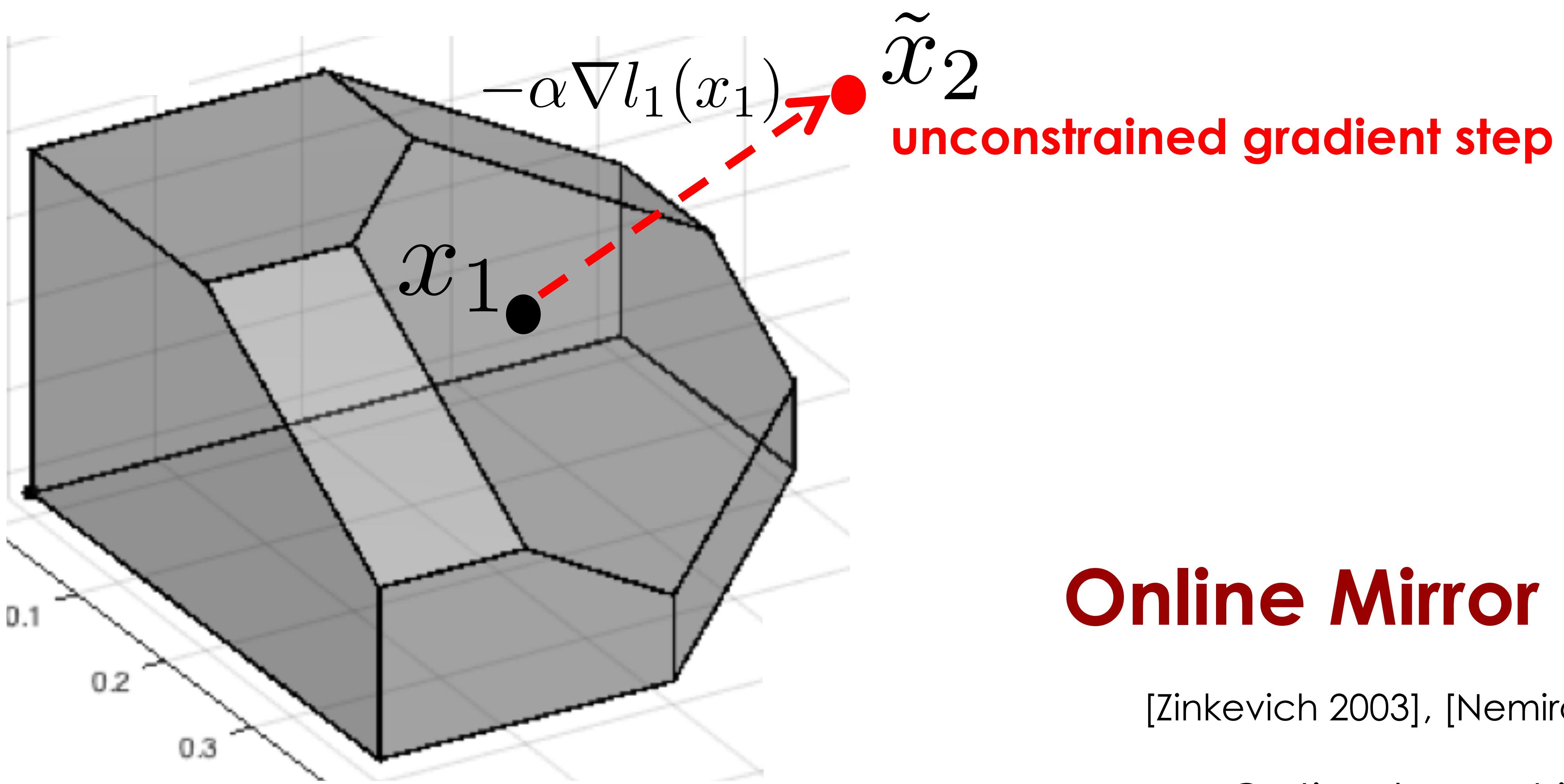
Online Mirror Descent

[Zinkevich 2003], [Nemirovski, Yudin 1983]

Optimal regret in many cases
[for e.g. Srebro, Sridharan, Tewari 2010]

*But **Computationally Slow**!*

Bottleneck in First-Order **Projection**-Based Algorithms



constrained decision set

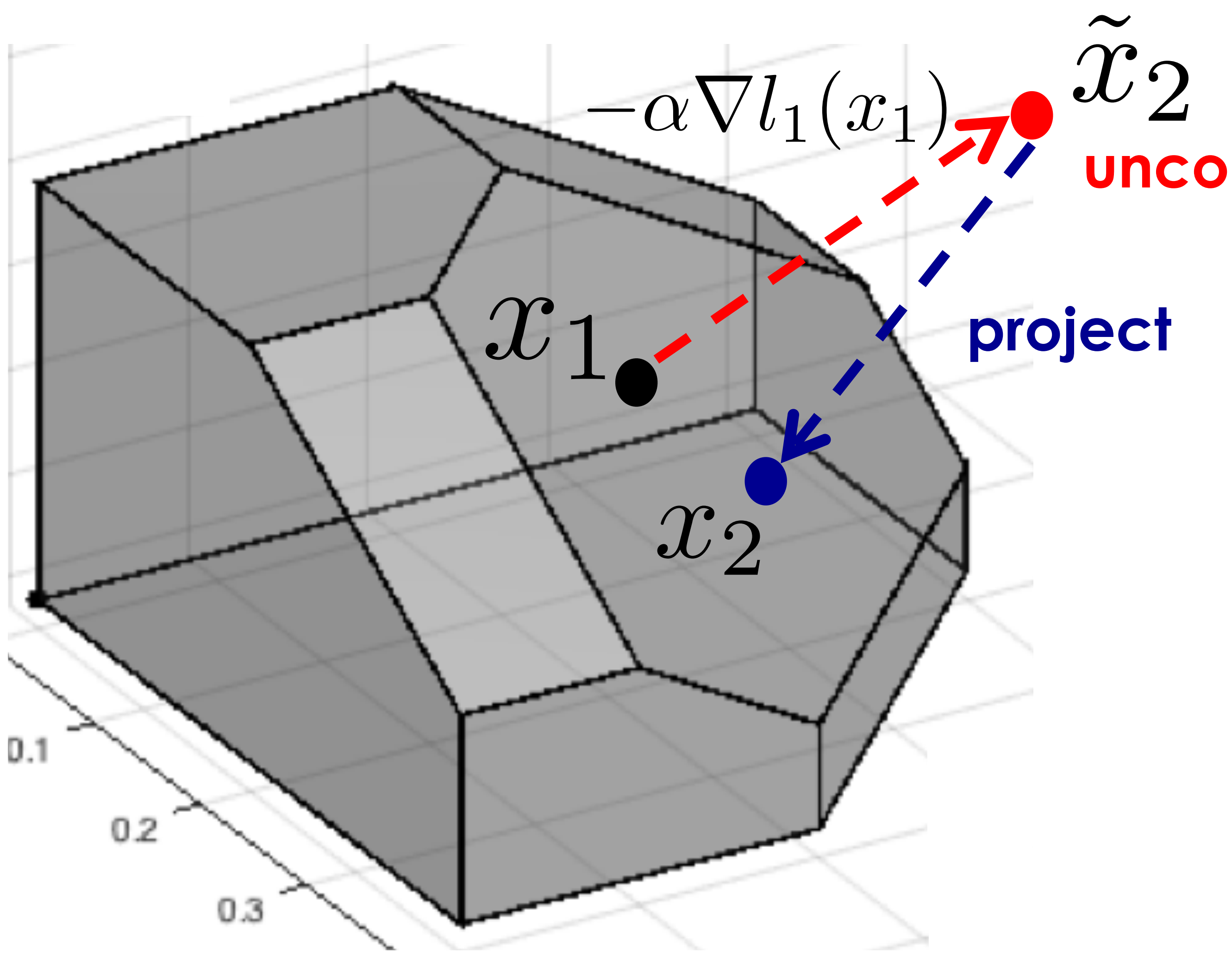
Online Mirror Descent

[Zinkevich 2003], [Nemirovski, Yudin 1983]

Optimal regret in many cases
[for e.g. Srebro, Sridharan, Tewari 2010]

*But **Computationally Slow**!*

Bottleneck in First-Order **Projection**-Based Algorithms



constrained decision set

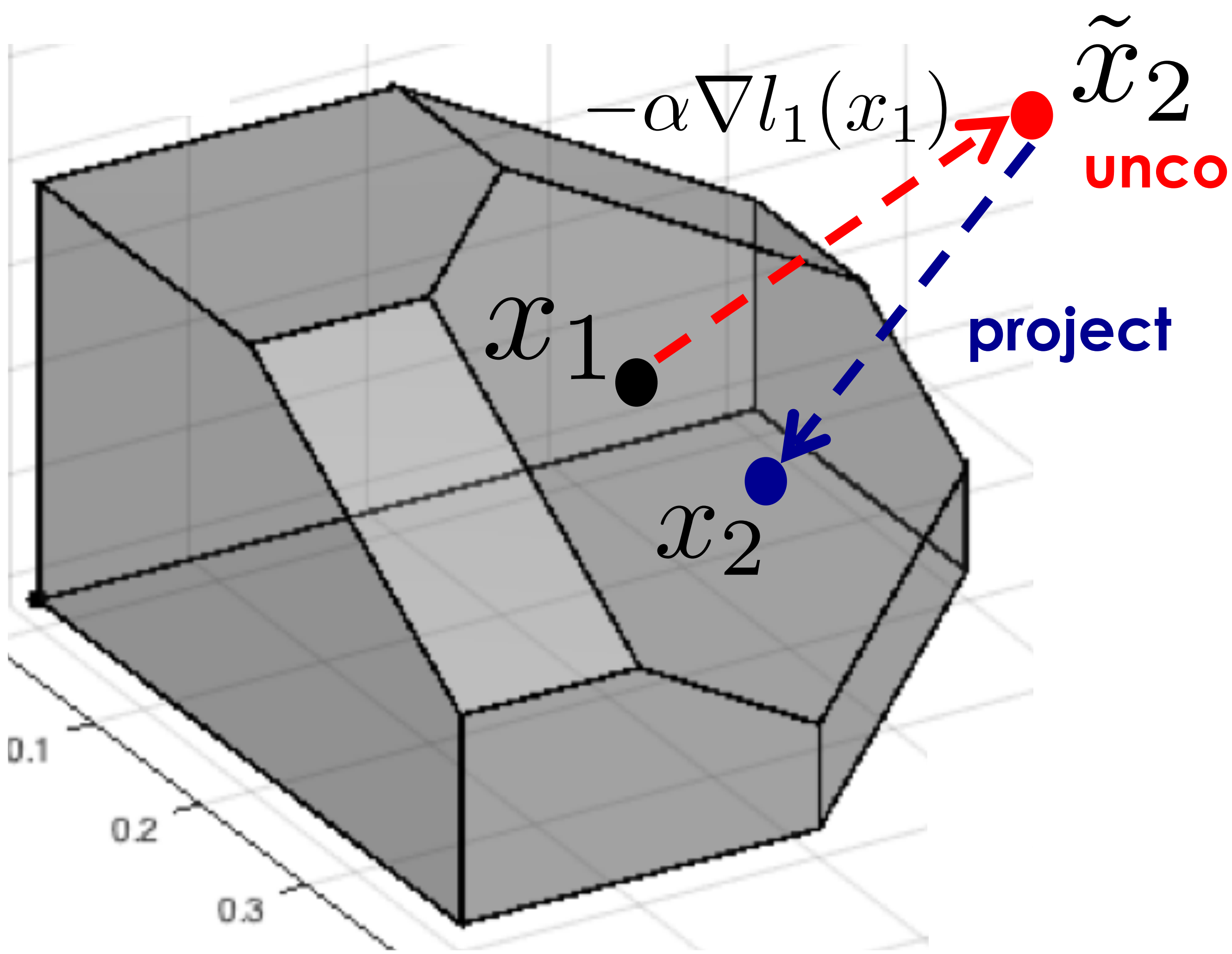
Online Mirror Descent

[Zinkevich 2003], [Nemirovski, Yudin 1983]

Optimal regret in many cases
[for e.g. Srebro, Sridharan, Tewari 2010]

*But **Computationally Slow**!*

Bottleneck in First-Order **Projection**-Based Algorithms



constrained decision set

Projections are obtained by **minimizing a convex function** (potentially in each time step)

Online Mirror Descent

[Zinkevich 2003], [Nemirovski, Yudin 1983]

Optimal regret in many cases
[for e.g. Srebro, Sridharan, Tewari 2010]

*But **Computationally Slow**!*

Outline

1. **Projections**
   - Motivation
   - **Problem setup**
   - **Novel algorithm**: Inc-Fix for separable convex minimization:
     - Main Result: O(n) SFM or O(n) Line searches
     - Exact computations, modulo solving a univariate equation

2. **Line Searches**
   - Previous best known: Megiddo's parametric search
   - Using Newton's Discrete Method: $n^2 + n \log^2 n$ SFM (n^6 improvement)

3. **What works best when**
   - Problems with Max-Cut and QUBO heuristics comparative studies
   - **Our framework**: Expanded instance library, Implementation of 37 heuristics, Large-scale cloud computing on the cross product
   - **Hyper-heuristic**: Map every instance to a feature space, learn "performance" of heuristics

(i) Which decision sets?

Submodular Base Polytopes

(i) Which decision sets?

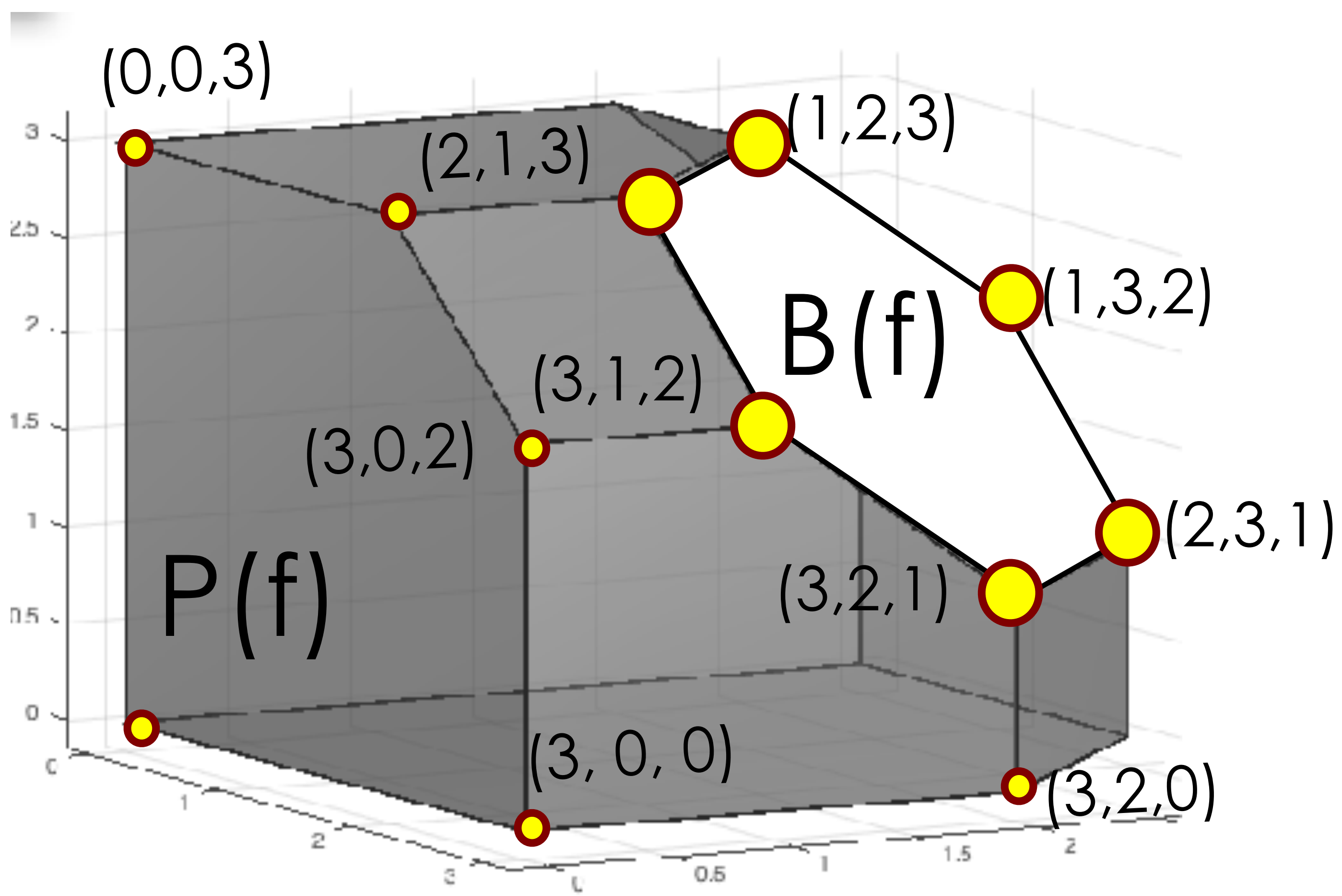
Submodular Base Polytopes

Permutations
1, 2, 3,
2, 3, 1,
3, 1, 2…

(i) Which decision sets?

Submodular Base Polytopes

Permutations
1, 2, 3,
2, 3, 1,
3, 1, 2…



(i) Which decision sets?

Submodular Base Polytopes

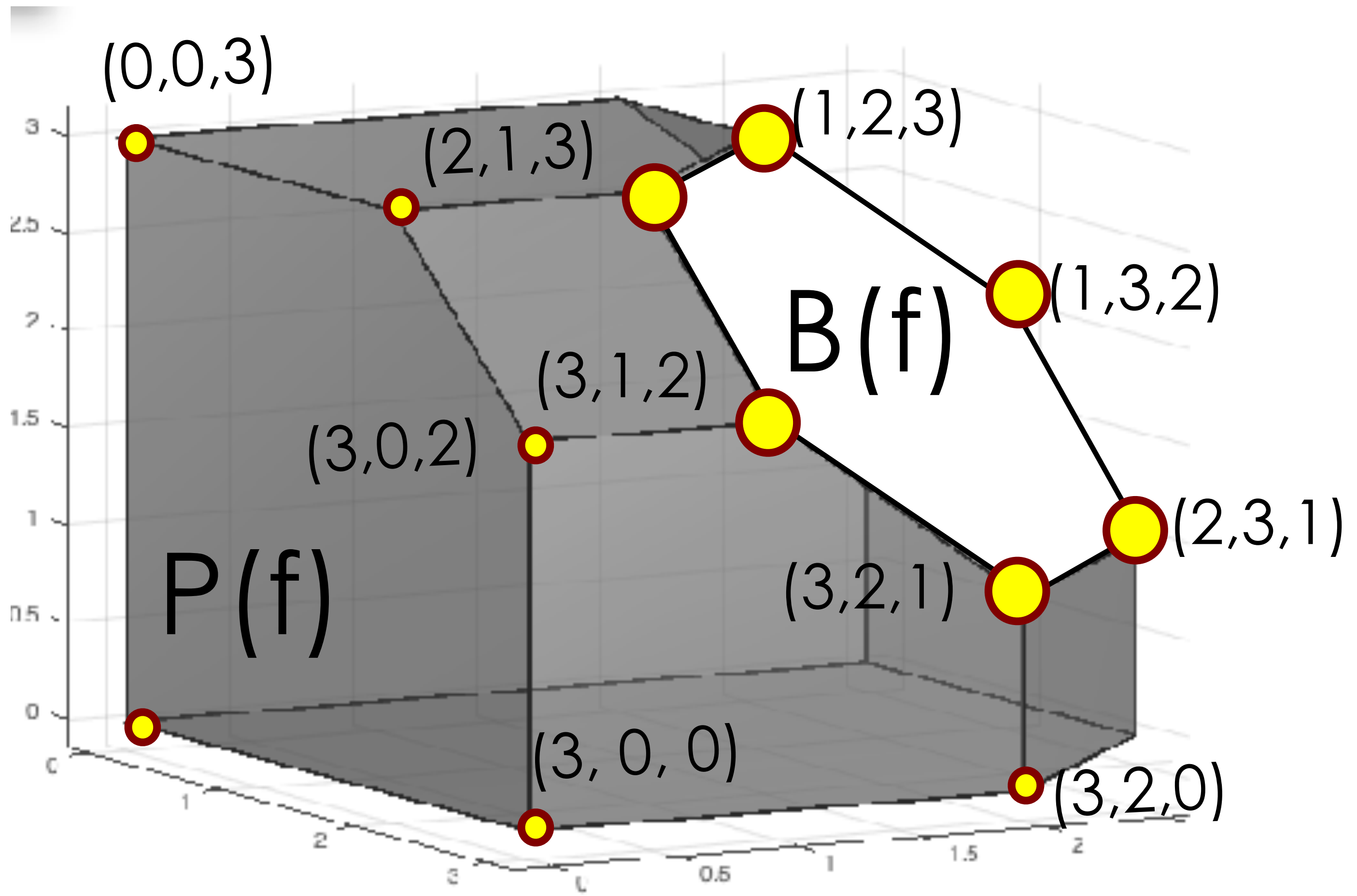
Permutations
1, 2, 3,
2, 3, 1,
3, 1, 2…

$$P(f) = \{x \in \mathbb{R}_+^E | \sum_{e \in S} x(e) \leq f(S) \; \forall S \subseteq E\}$$

$$B(f) = \{x \in P(f) | \sum_{e \in E} x(e) = f(E)\}$$

Ground set **E**

Submodular set function
Captures the property of diminishing returns



(i) Which decision sets?

Submodular Base Polytopes

Permutations
1, 2, 3,
2, 3, 1,
3, 1, 2…

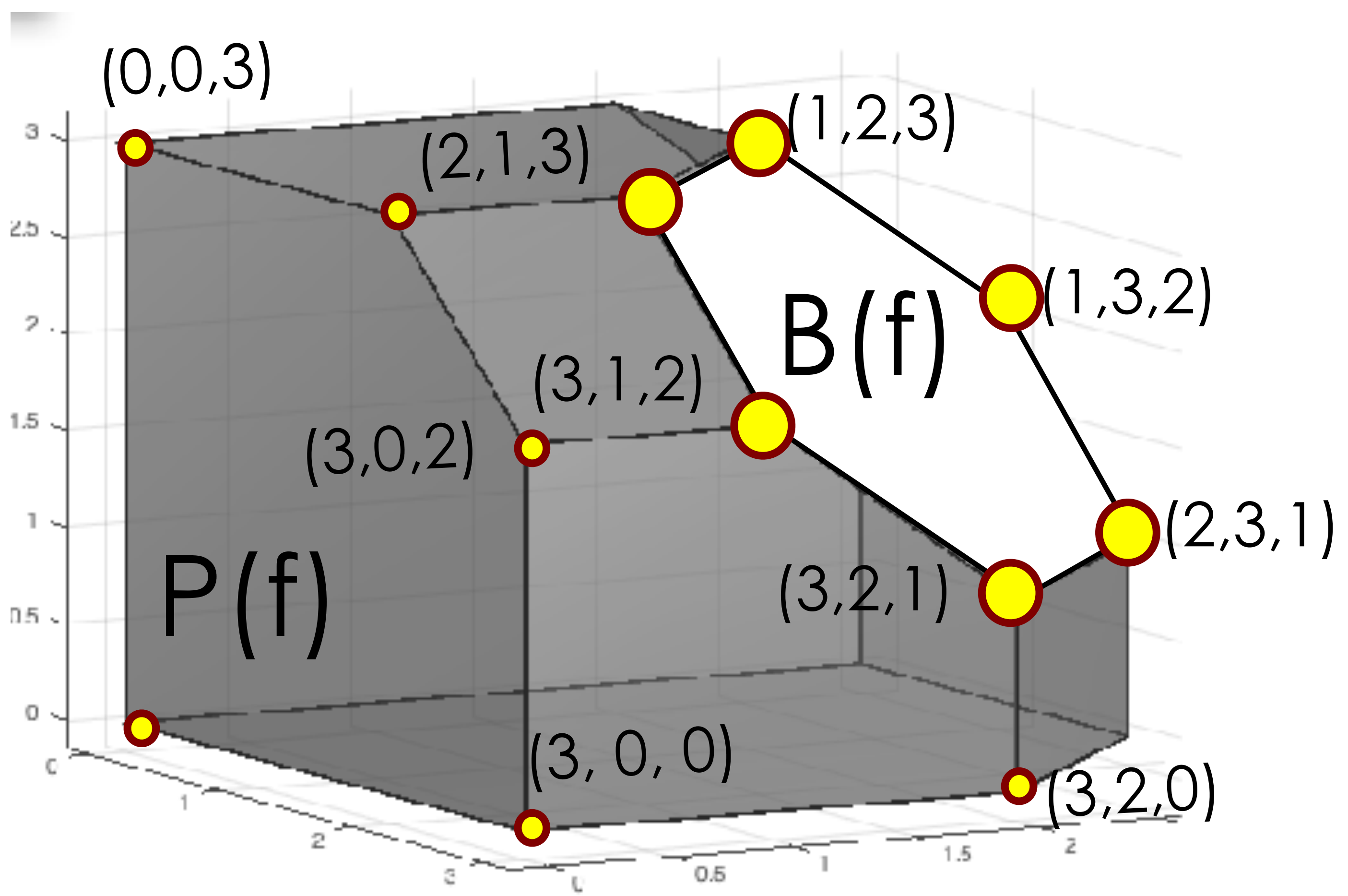
$$P(f) = \{x \in \mathbb{R}_+^E | \sum_{e \in S} x(e) \leq f(S) \ \forall S \subseteq E\}$$

Exp!

$$B(f) = \{x \in P(f) | \sum_{e \in E} x(e) = f(E)\}$$

Ground set **E**

Submodular set function
Captures the property of diminishing returns



(i) Which decision sets?

Submodular Base Polytopes

Permutations
1, 2, 3,
2, 3, 1,
3, 1, 2…

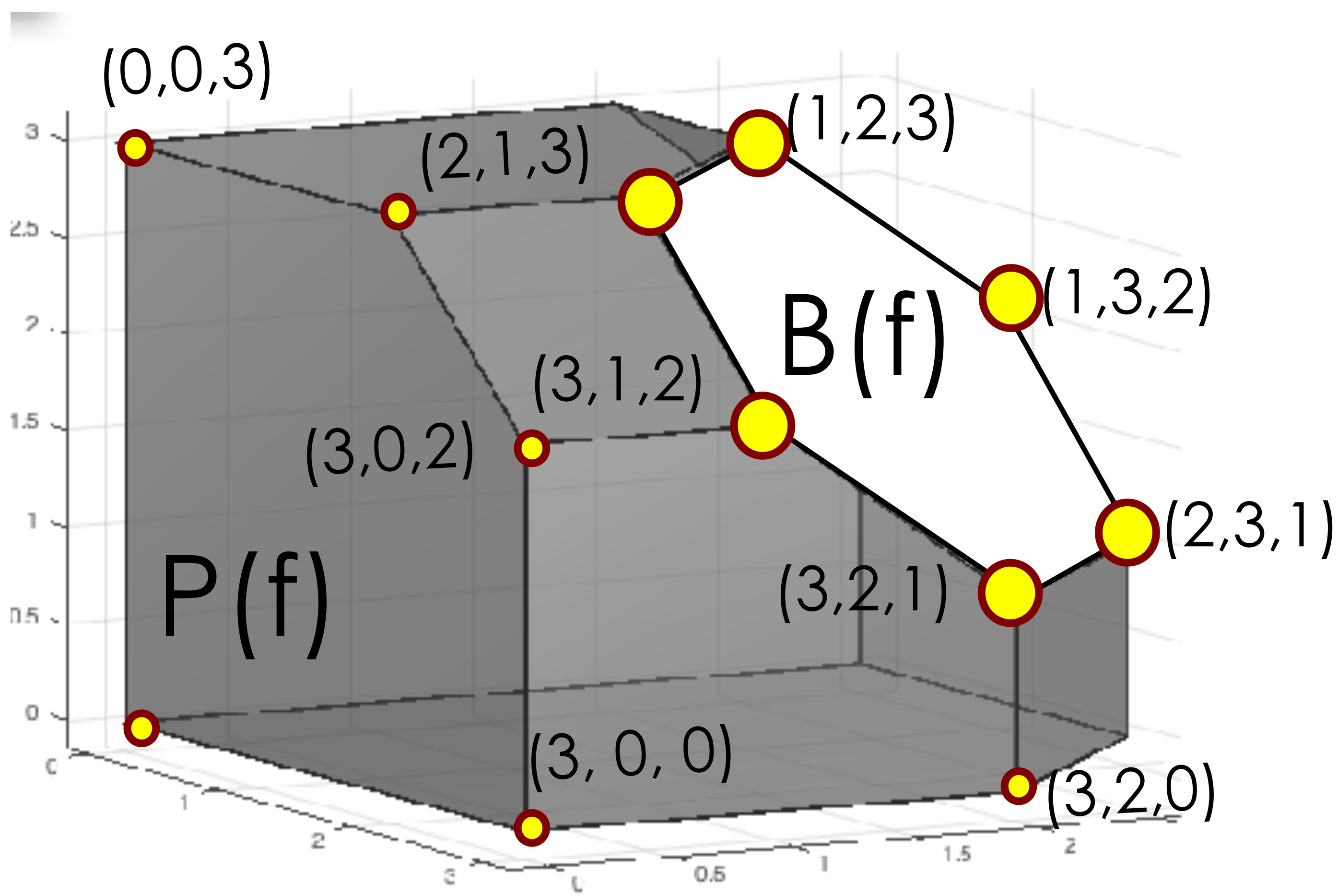
$$P(f) = \{x \in \mathbb{R}_+^E | \sum_{e \in S} x(e) \leq f(S)\ \forall S \subseteq E\}$$

Exp!

$$B(f) = \{x \in P(f) | \sum_{e \in E} x(e) = f(E)\}$$

Ground set **E**

Submodular set function
Captures the property of diminishing returns



Choice of **f(.)** gives different structures

f(S) = $\sum_{s=1}^{|S|} (n - s + 1)$

(i) Which decision sets?

Submodular Base Polytopes

Permutations
1, 2, 3,
2, 3, 1,
3, 1, 2…

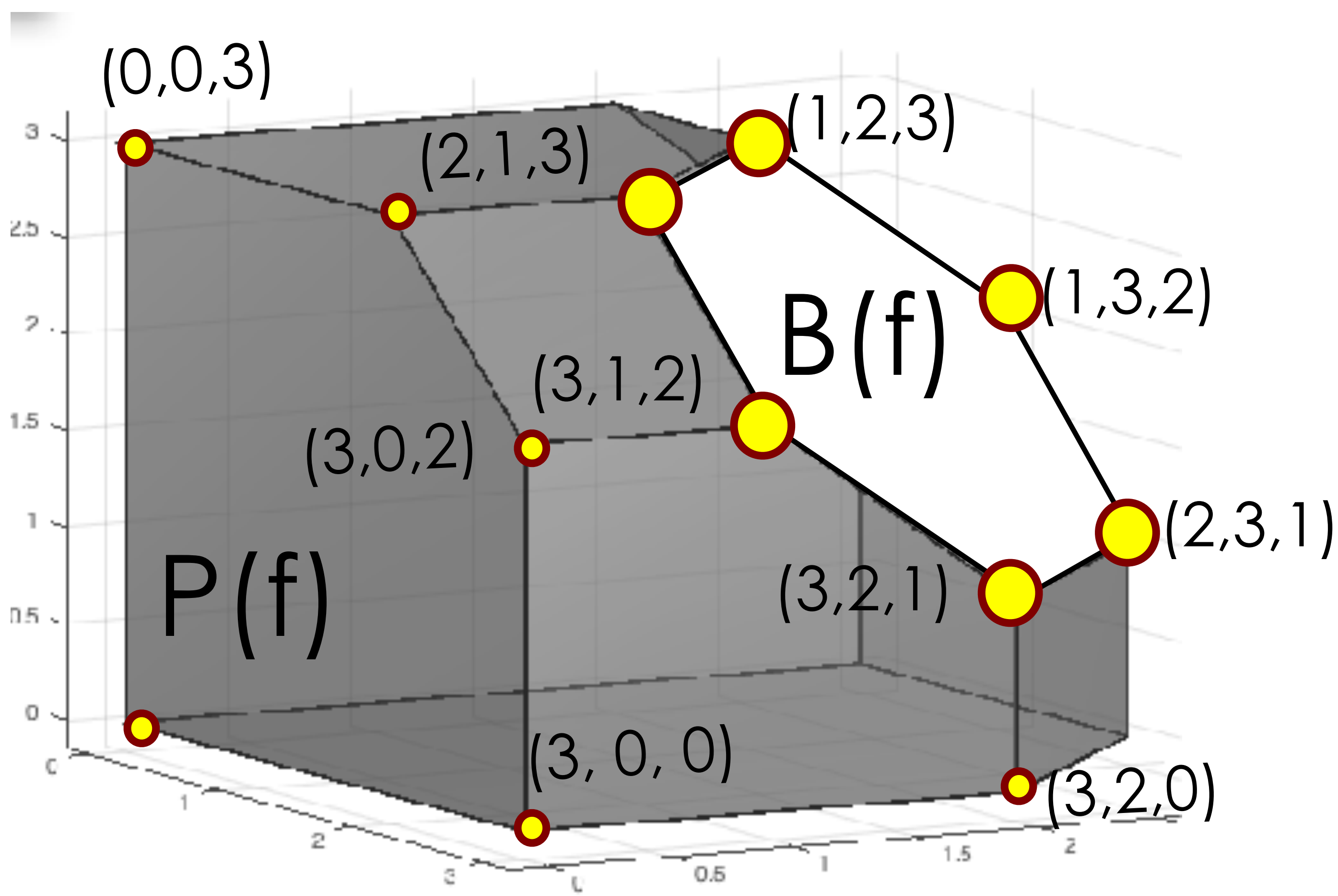
$$P(f) = \{x \in \mathbb{R}_+^E | \sum_{e \in S} x(e) \leq f(S)\ \forall S \subseteq E\}$$

Exp!

$$B(f) = \{x \in P(f) | \sum_{e \in E} x(e) = f(E)\}$$

Ground set **E**

Submodular set function
Captures the property of diminishing returns



Choice of **f(.)** gives different structures

f(S) = $\sum_{s=1}^{|S|} (n - s + 1)$

MANY MANY MORE INTERESTING EXAMPLES!!

(ii) Minimize what?

- Bregman Divergences

$$D_\omega(x,y) := \omega(x) - \omega(y) - \nabla\omega(y)^T(x-y)$$

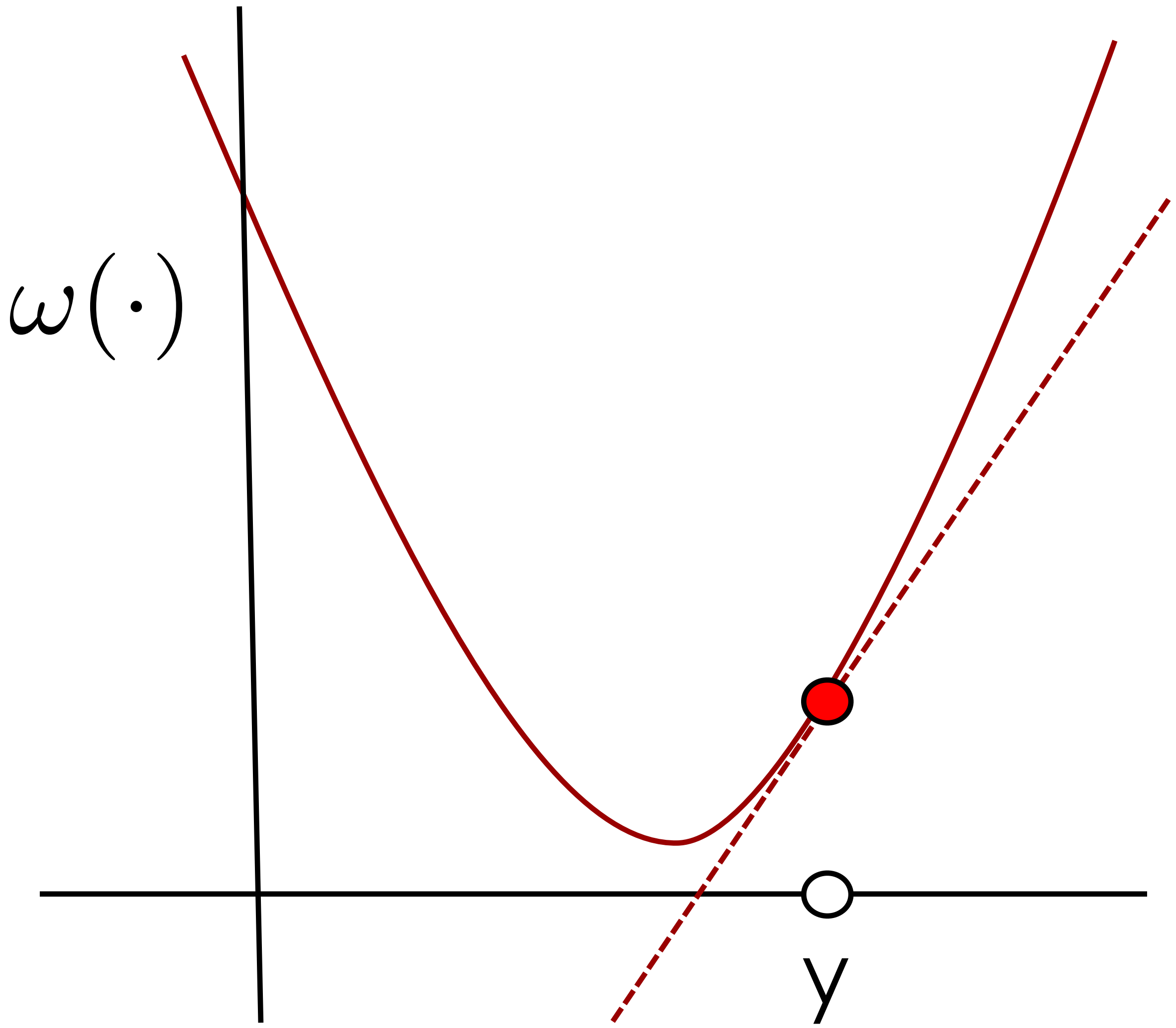
Convex, non-negative, not symmetric

(ii) Minimize what?

- Bregman Divergences

$$D_\omega(x, y) := \omega(x) - \omega(y) - \nabla\omega(y)^T(x - y)$$

Convex, non-negative, not symmetric

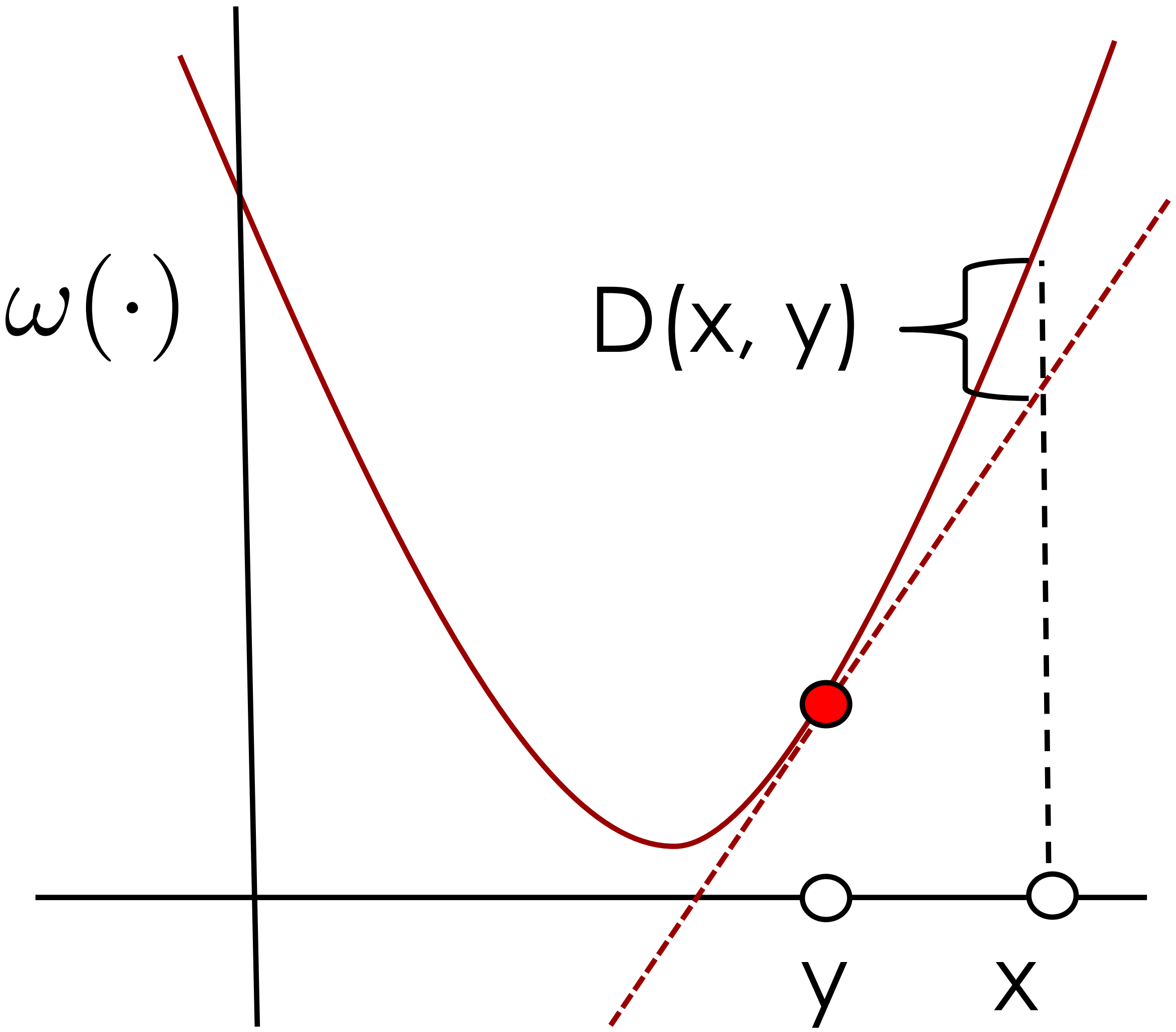


(ii) Minimize what?

- Bregman Divergences

$$D_\omega(x, y) := \omega(x) - \omega(y) - \nabla\omega(y)^T(x - y)$$

Convex, non-negative, not symmetric

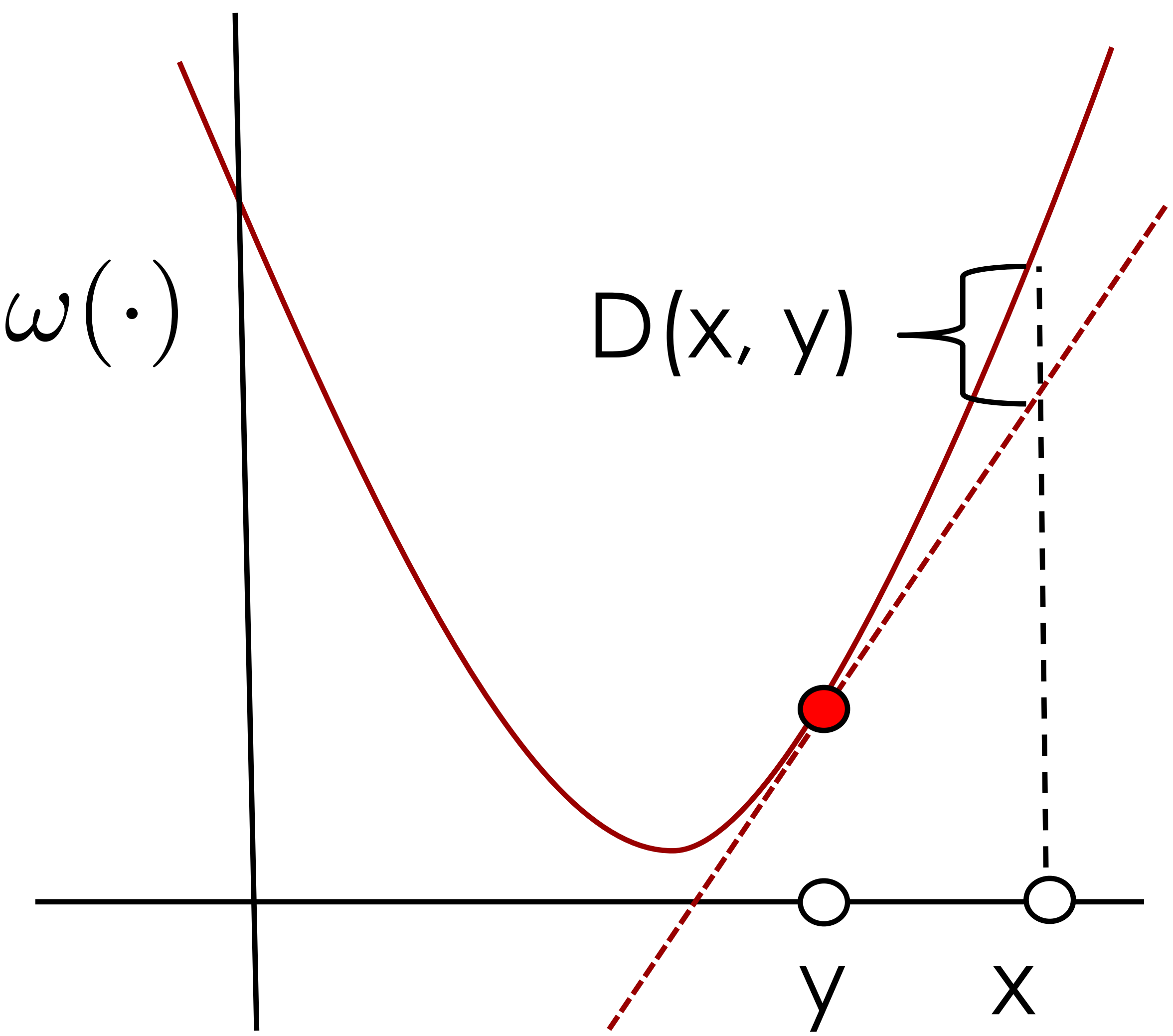


(ii) Minimize what?

- Bregman Divergences

$$D_\omega(x, y) := \omega(x) - \omega(y) - \nabla\omega(y)^T(x - y)$$

Convex, non-negative, not symmetric



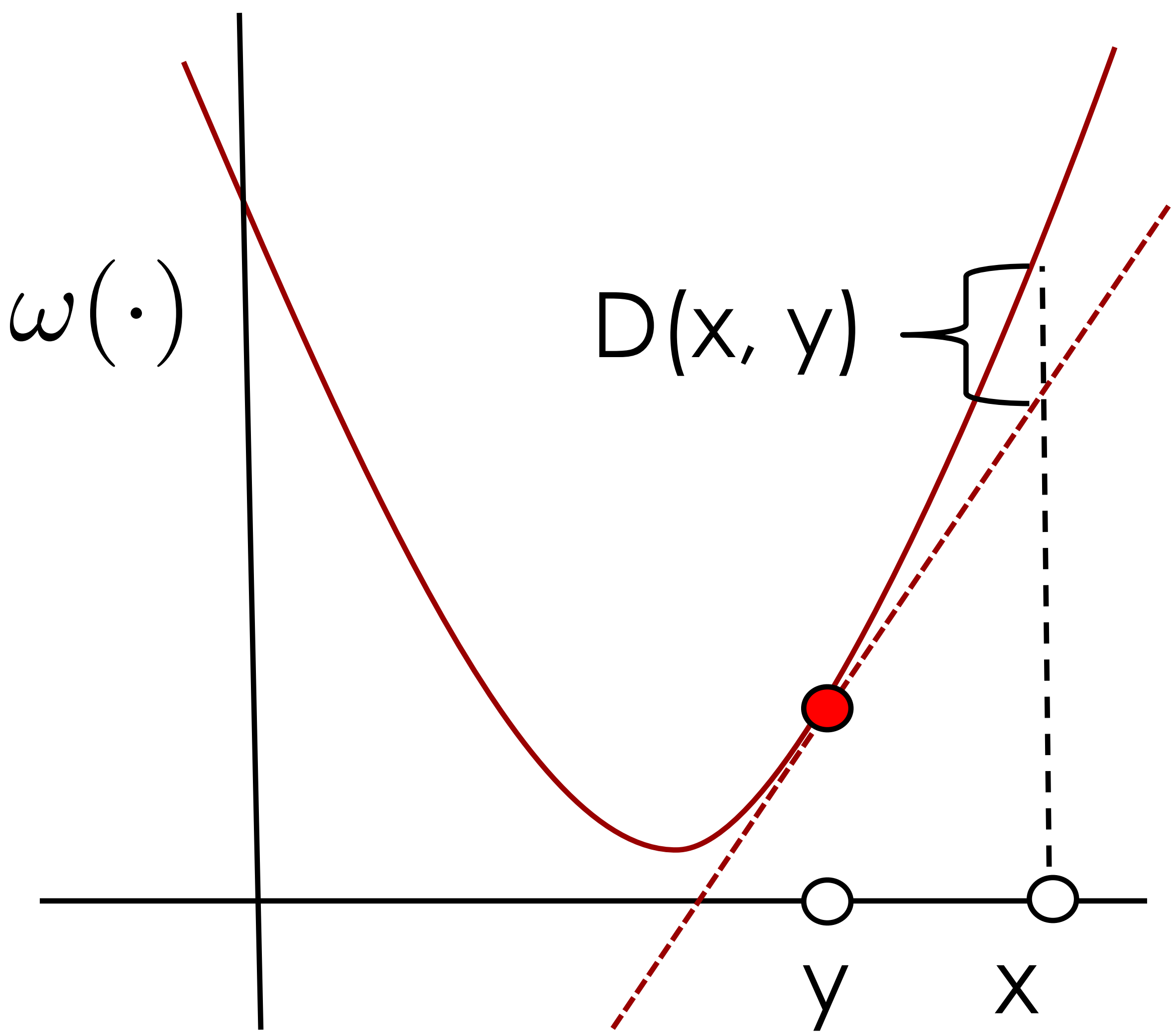
$\omega(\cdot)$	$D_\omega(x, y)$

(ii) Minimize what?

- Bregman Divergences

$$D_\omega(x,y) := \omega(x) - \omega(y) - \nabla\omega(y)^T(x-y)$$

Convex, non-negative, not symmetric



$\omega(\cdot)$	$D_\omega(x,y)$
$\frac{1}{2}\|x\|^2$	$\frac{1}{2}\|x-y\|^2$
$\sum_e x(e)\ln x(e) - x(e)$	$\sum_{e\in E} x(e)\ln\frac{x(e)}{y(e)} - x(e) + y(e)$
$-\sum_e \ln x(e)$	$\sum_e \frac{x(e)}{y(e)} - \ln\frac{x(e)}{y(e)} - 1$

(ii) Minimize what?

- Bregman Divergences

$$D_\omega(x, y) := \omega(x) - \omega(y) - \nabla\omega(y)^T(x - y)$$

Convex, non-negative, not symmetric

$\omega(\cdot)$	$D_\omega(x, y)$
$\frac{1}{2}\|x\|^2$	$\frac{1}{2}\|x - y\|^2$
$\sum_e x(e) \ln x(e) - x(e)$	$\sum_{e \in E} x(e) \ln \frac{x(e)}{y(e)} - x(e) + y(e)$
$-\sum_e \ln x(e)$	$\sum_e \frac{x(e)}{y(e)} - \ln \frac{x(e)}{y(e)} - 1$

Separable Strictly Convex Functions

$$\min_{x \in B(f)} h(x) := \sum_{e \in E} h_e(x(e))$$

(ii) Minimize what?

- Bregman Divergences

$$D_\omega(x,y) := \omega(x) - \omega(y) - \nabla\omega(y)^T(x-y)$$

Convex, non-negative, not symmetric

Separable Strictly Convex Functions

$$\min_{x \in B(f)} h(x) := \sum_{e \in E} h_e(x(e))$$

$\omega(\cdot)$	$D_\omega(x,y)$
$\frac{1}{2}\|x\|^2$	$\frac{1}{2}\|x-y\|^2$
$\sum_e x(e)\ln x(e) - x(e)$	$\sum_{e\in E} x(e)\ln\frac{x(e)}{y(e)} - x(e) + y(e)$
$-\sum_e \ln x(e)$	$\sum_e \frac{x(e)}{y(e)} - \ln\frac{x(e)}{y(e)} - 1$

Why do we need different divergences: convergence, regret bounds

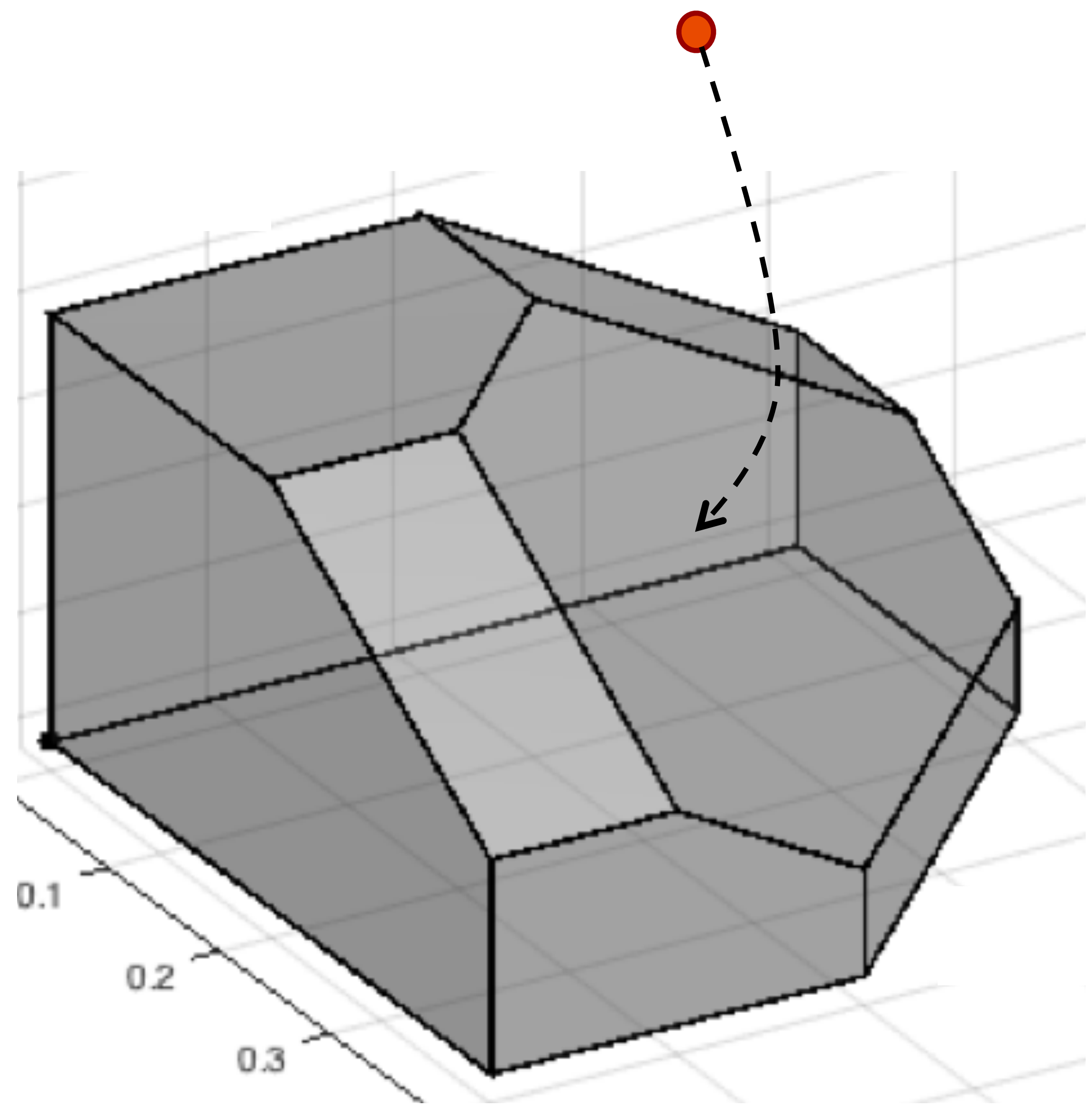
Algorithm Inc-Fix

For Separable Strictly Convex Minimization Over Base Polytopes:

$$\min_{x \in B(f)} \sum_{e \in E} h_e(x(e))$$

(a). Which decision sets?
Submodular Base Polytopes: B(f)
(Permutations, k-subsets..)

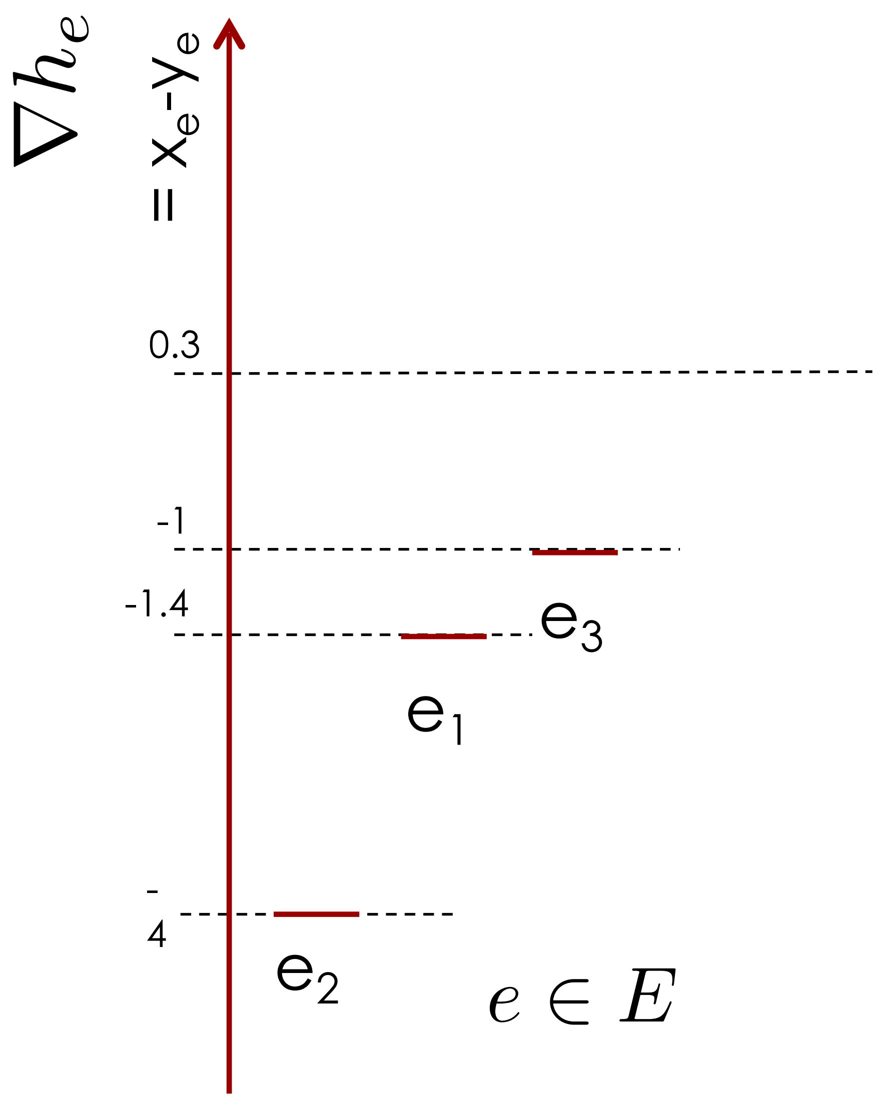
(b). Minimizing separable convex fns
(sq. Euclidean distance, KL-divergence, ...)

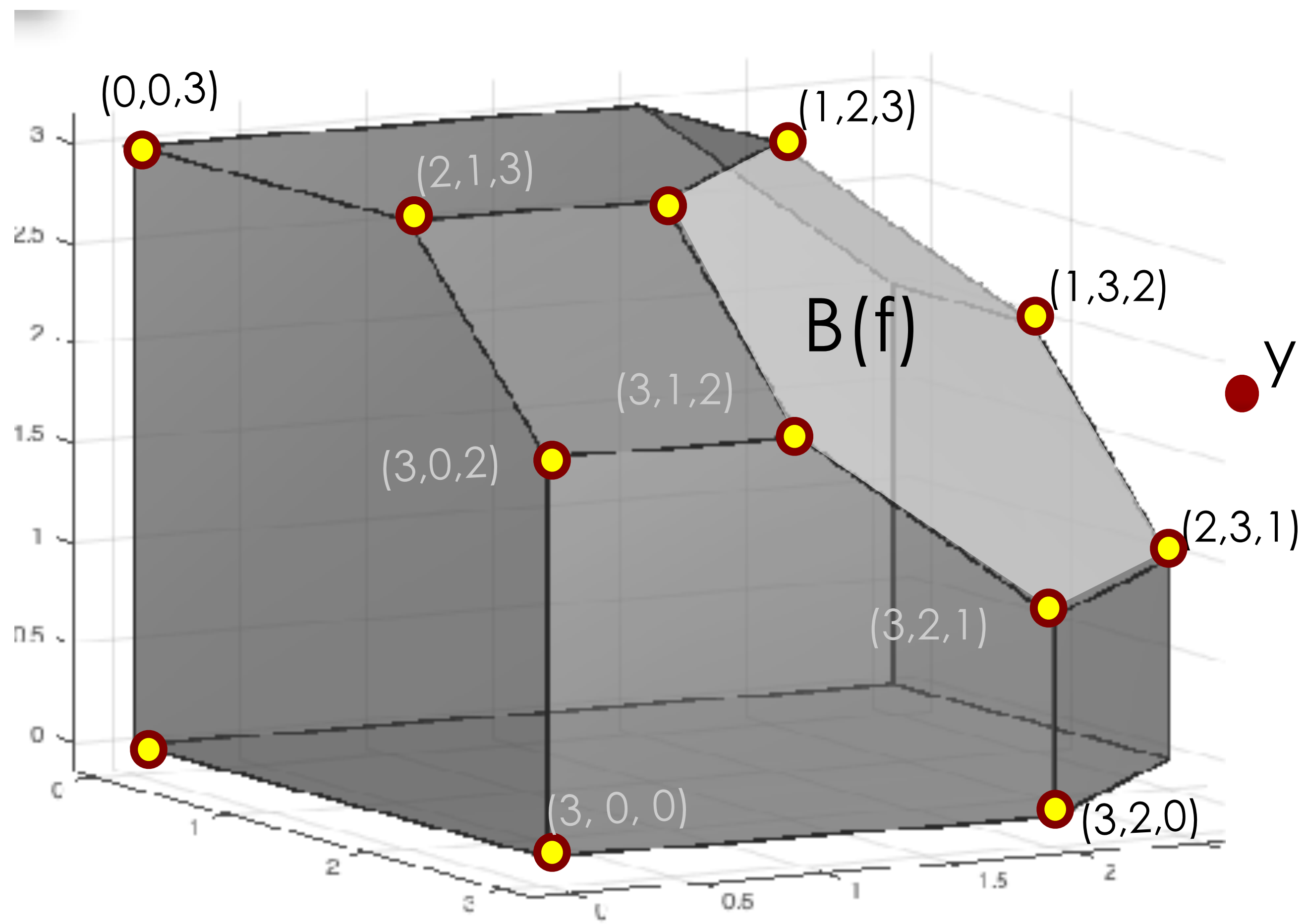


Inc-Fix Algorithm

Project: y = (1.4, 4, 1)ᵀ
under Euclidean distance

$$\min_{x \in B(f)} \frac{1}{2} \sum_e (x(e) - y(e))^2$$





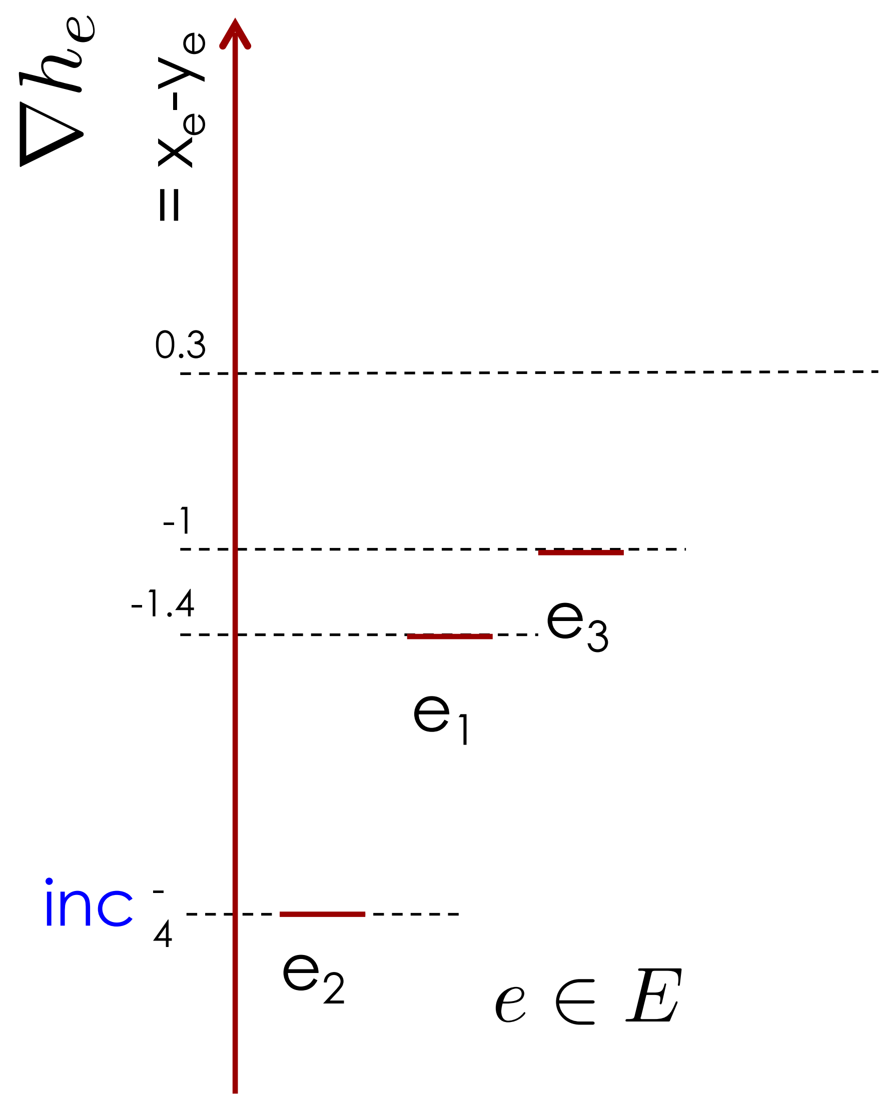
"greedy in gradient space" – proof from first-order optimality conditions

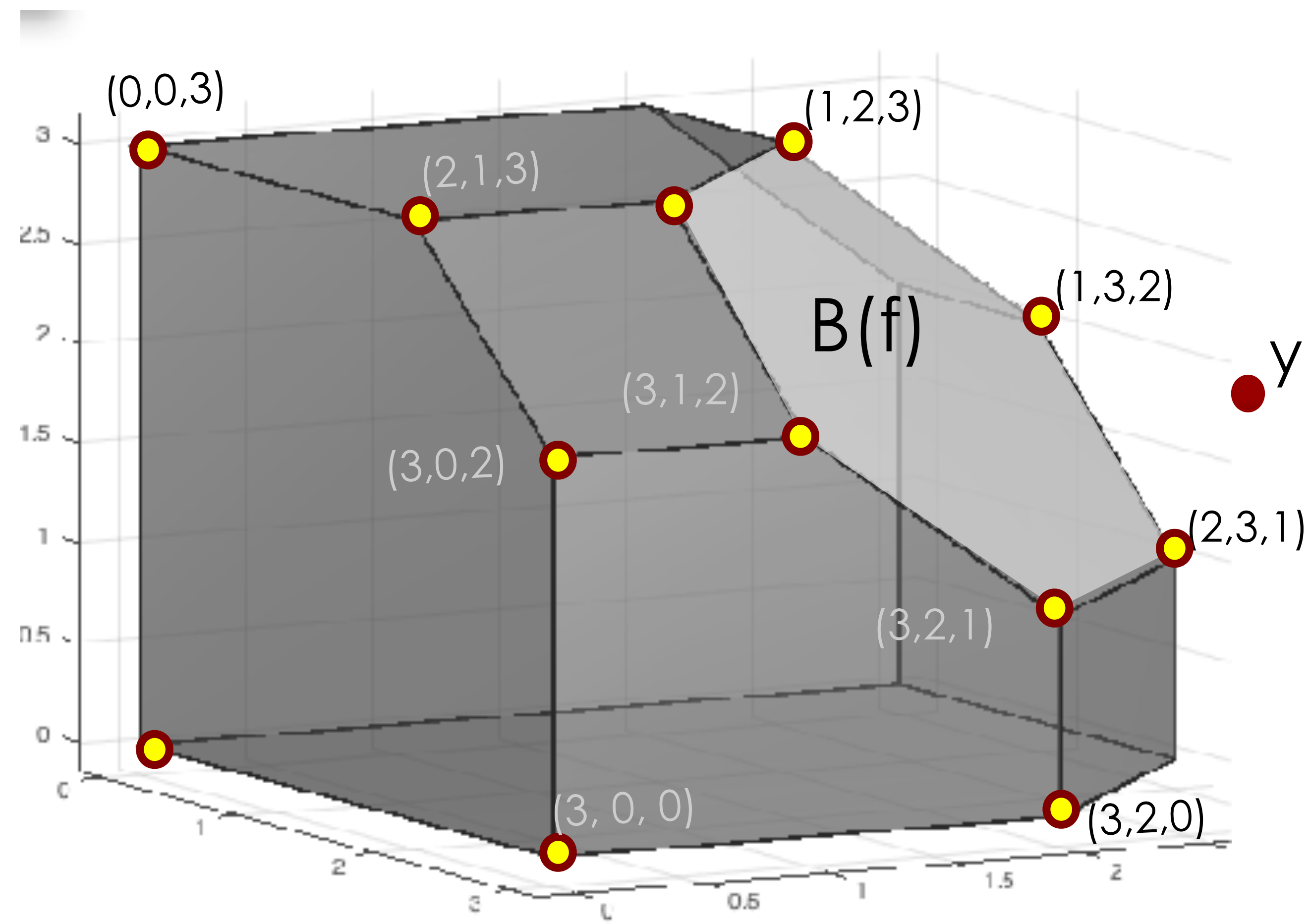
[Gupta, Goemans, Jaillet]

Inc-Fix Algorithm

Project: y = (1.4, 4, 1)^T
under Euclidean distance

$$\min_{x \in B(f)} \frac{1}{2} \sum_e (x(e) - y(e))^2$$





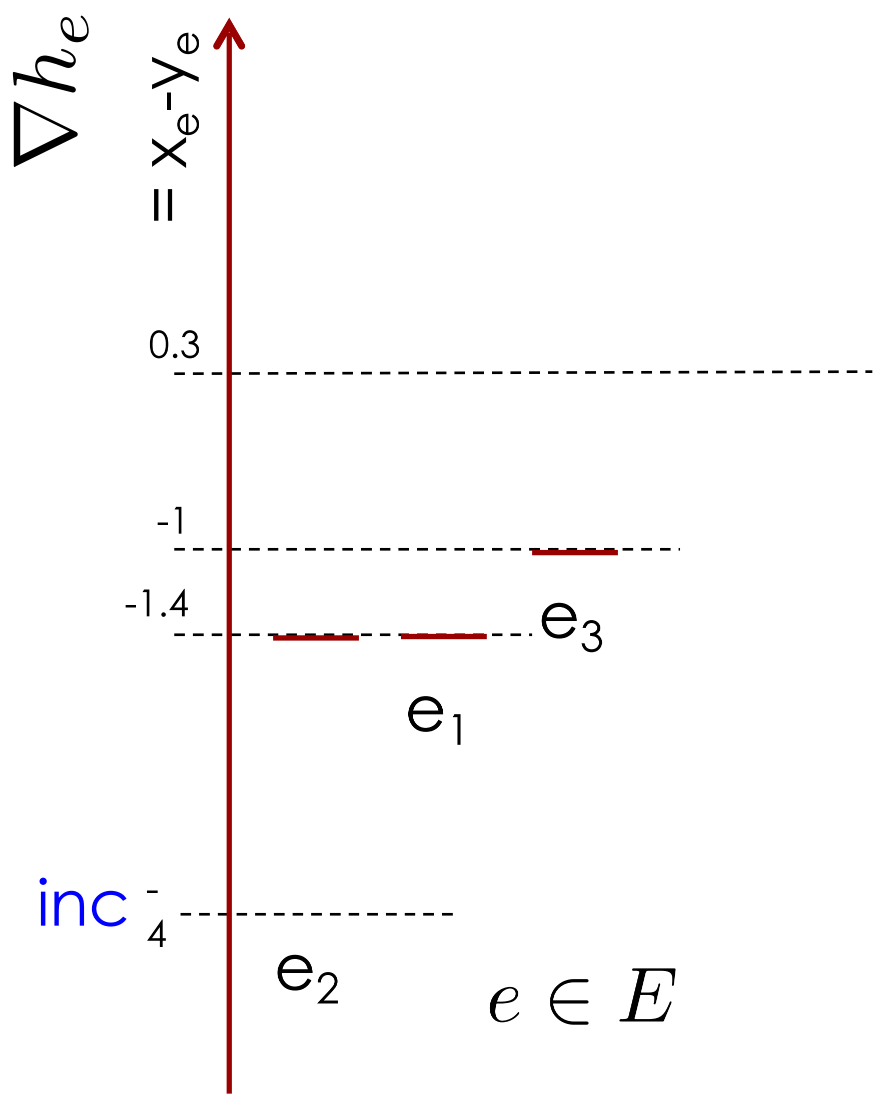
“greedy in gradient space” – proof from first-order optimality conditions

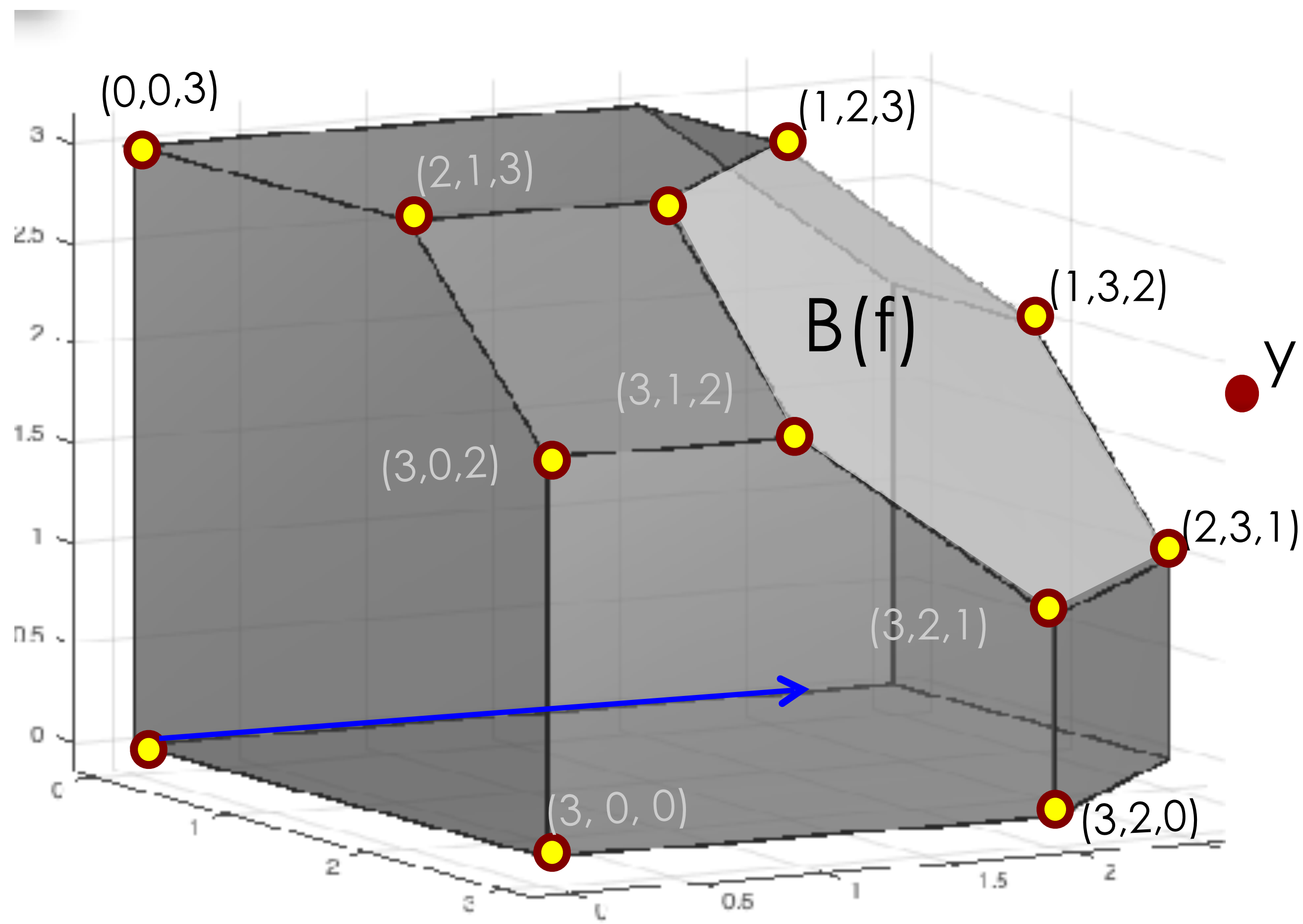
[Gupta, Goemans, Jaillet]

Inc-Fix Algorithm

Project: **y = (1.4, 4, 1)ᵀ** under Euclidean distance

$$\min_{x \in B(f)} \frac{1}{2} \sum_e (x(e) - y(e))^2$$





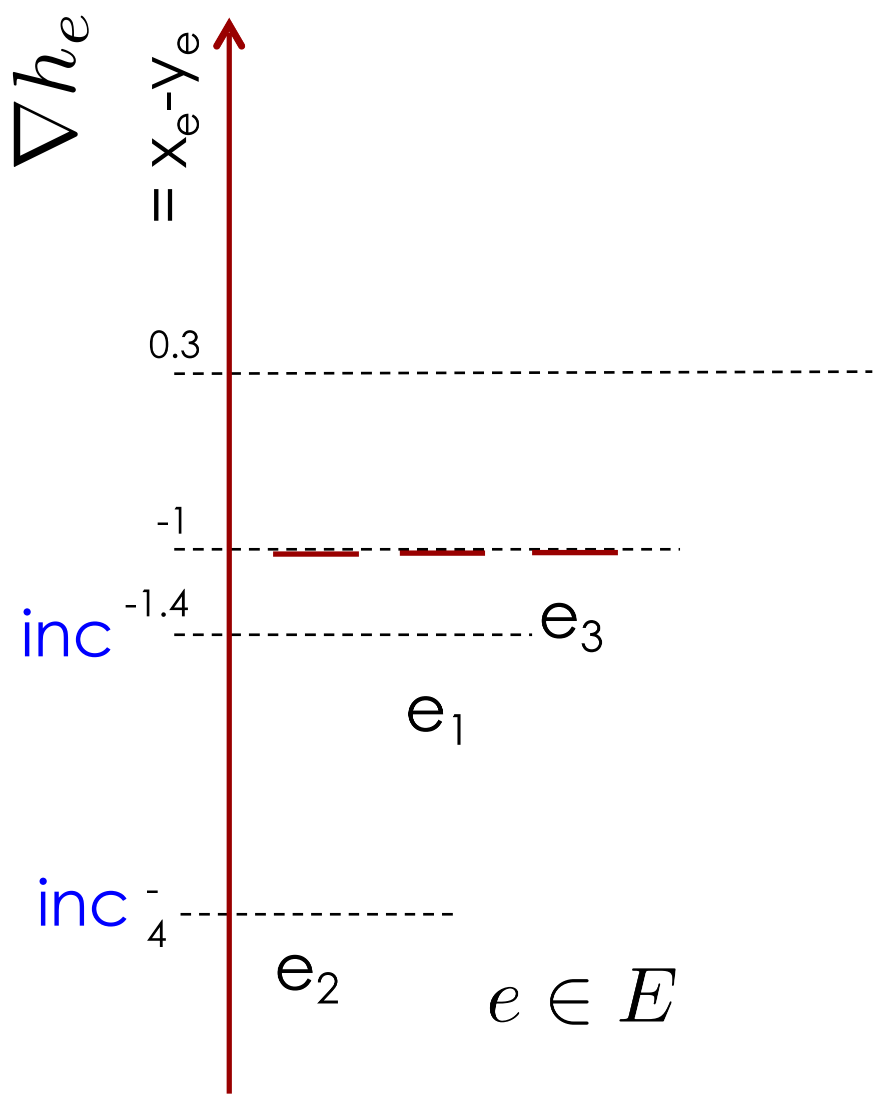
"greedy in gradient space" – proof from first-order optimality conditions

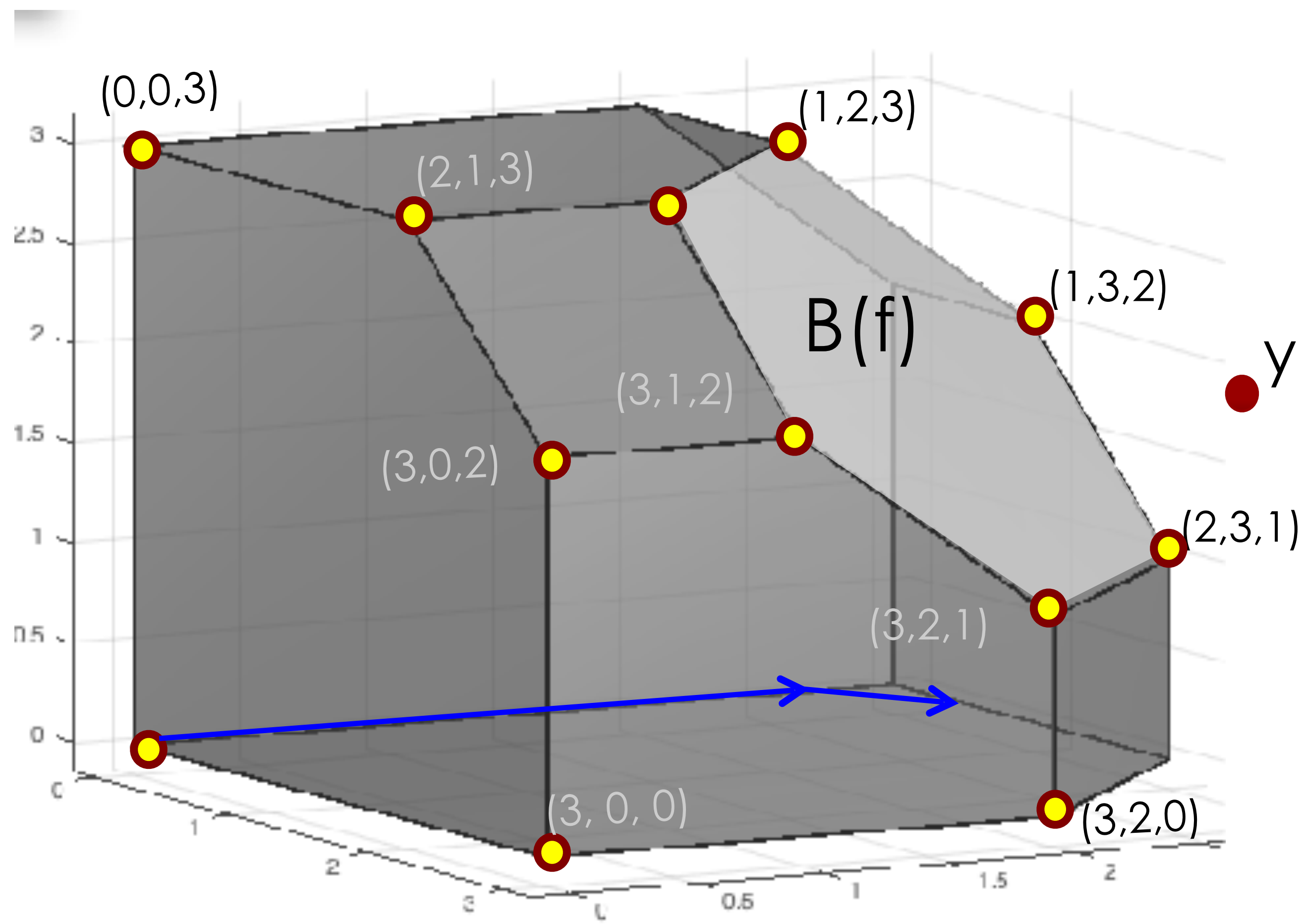
[Gupta, Goemans, Jaillet]

Inc-Fix Algorithm

Project: **y = (1.4, 4, 1)ᵀ** under Euclidean distance

$$\min_{x \in B(f)} \frac{1}{2} \sum_e (x(e) - y(e))^2$$





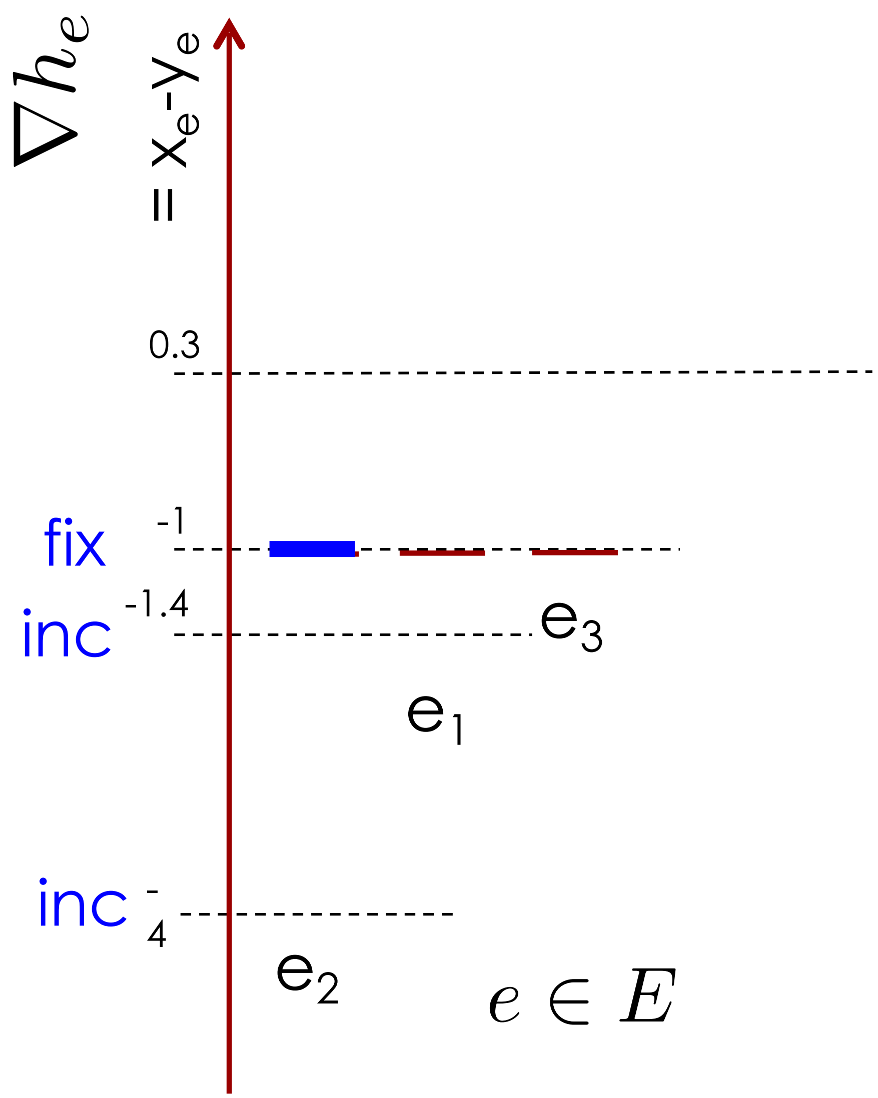
"greedy in gradient space" – proof from first-order optimality conditions

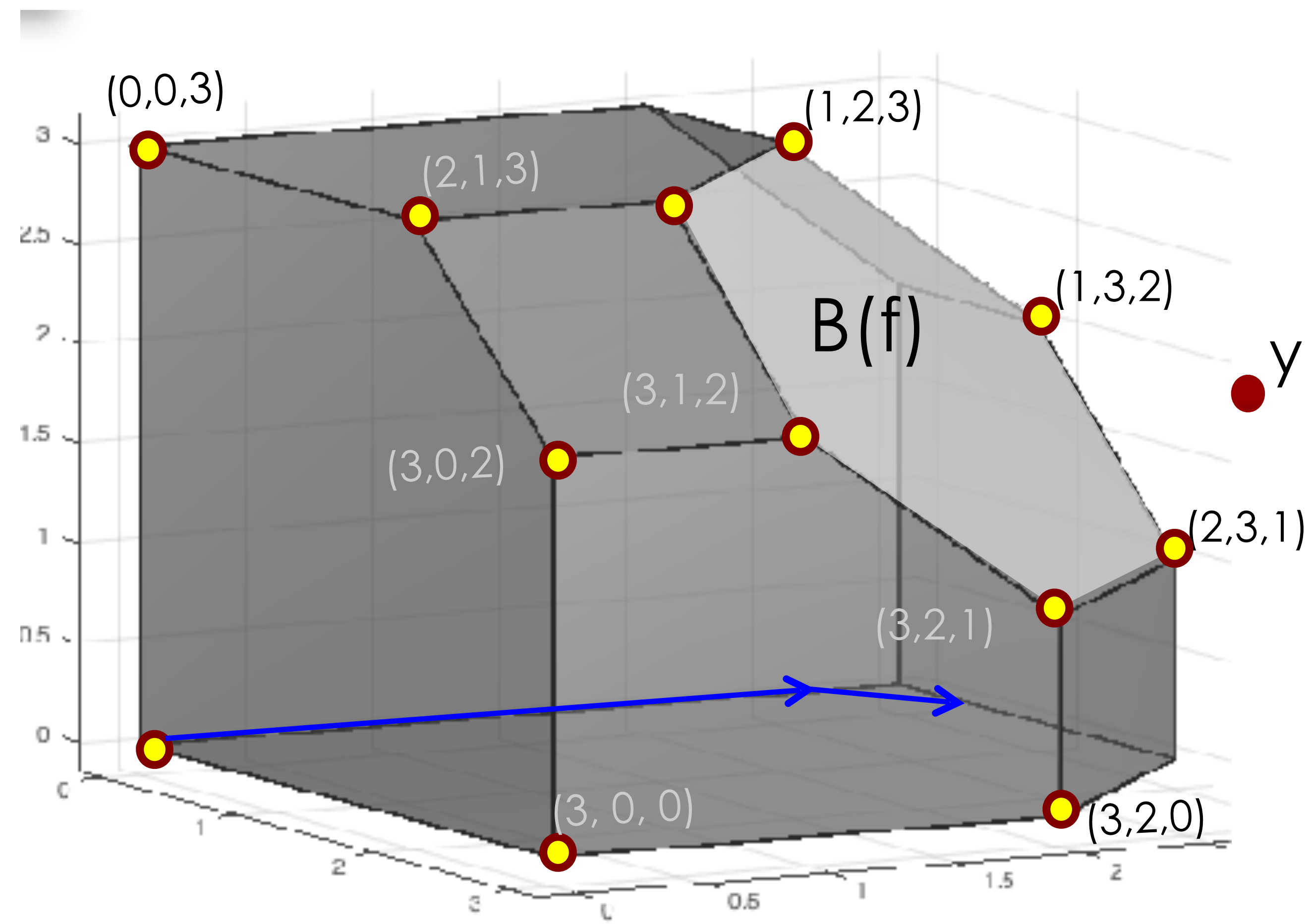
[Gupta, Goemans, Jaillet]

Inc-Fix Algorithm

Project: **y = (1.4, 4, 1)^T**
under Euclidean distance

$$\min_{x \in B(f)} \frac{1}{2} \sum_e (x(e) - y(e))^2$$





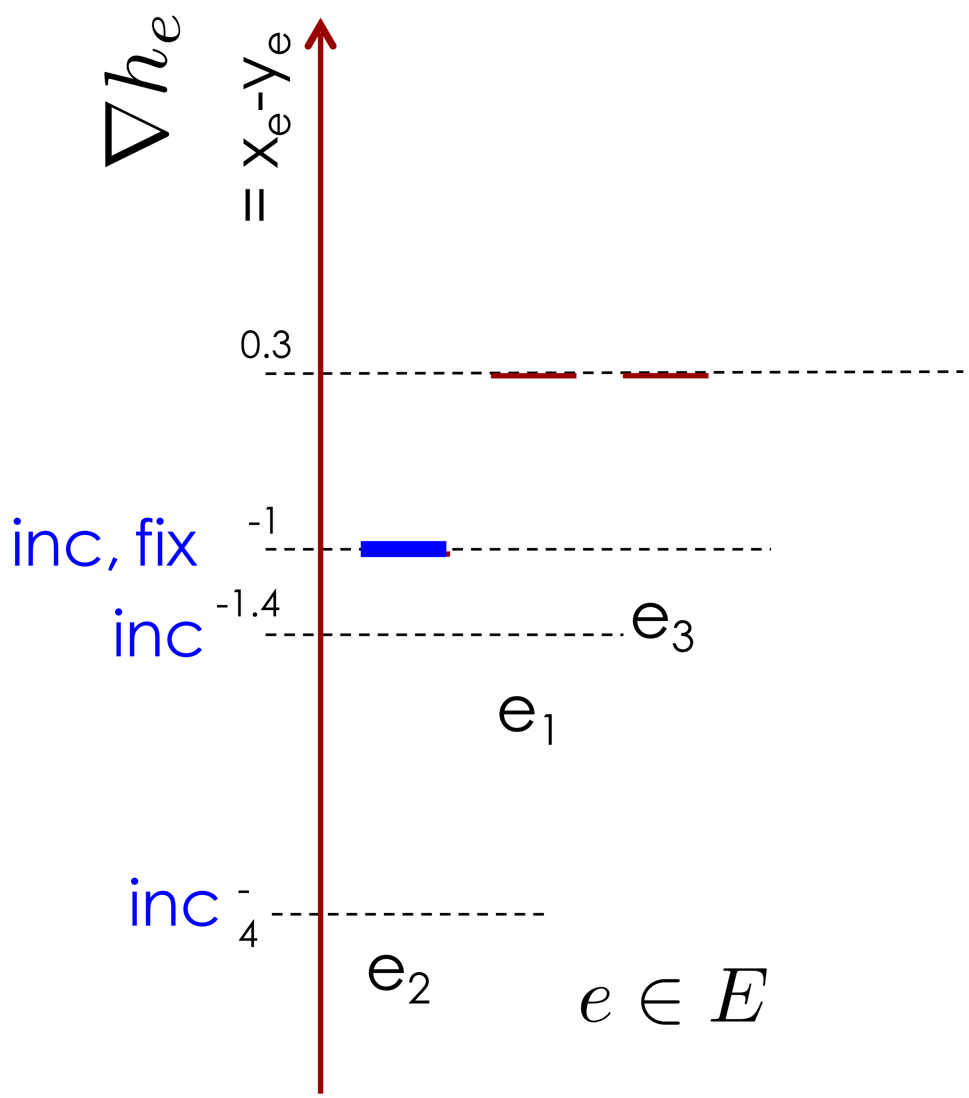
"greedy in gradient space" – proof from first-order optimality conditions

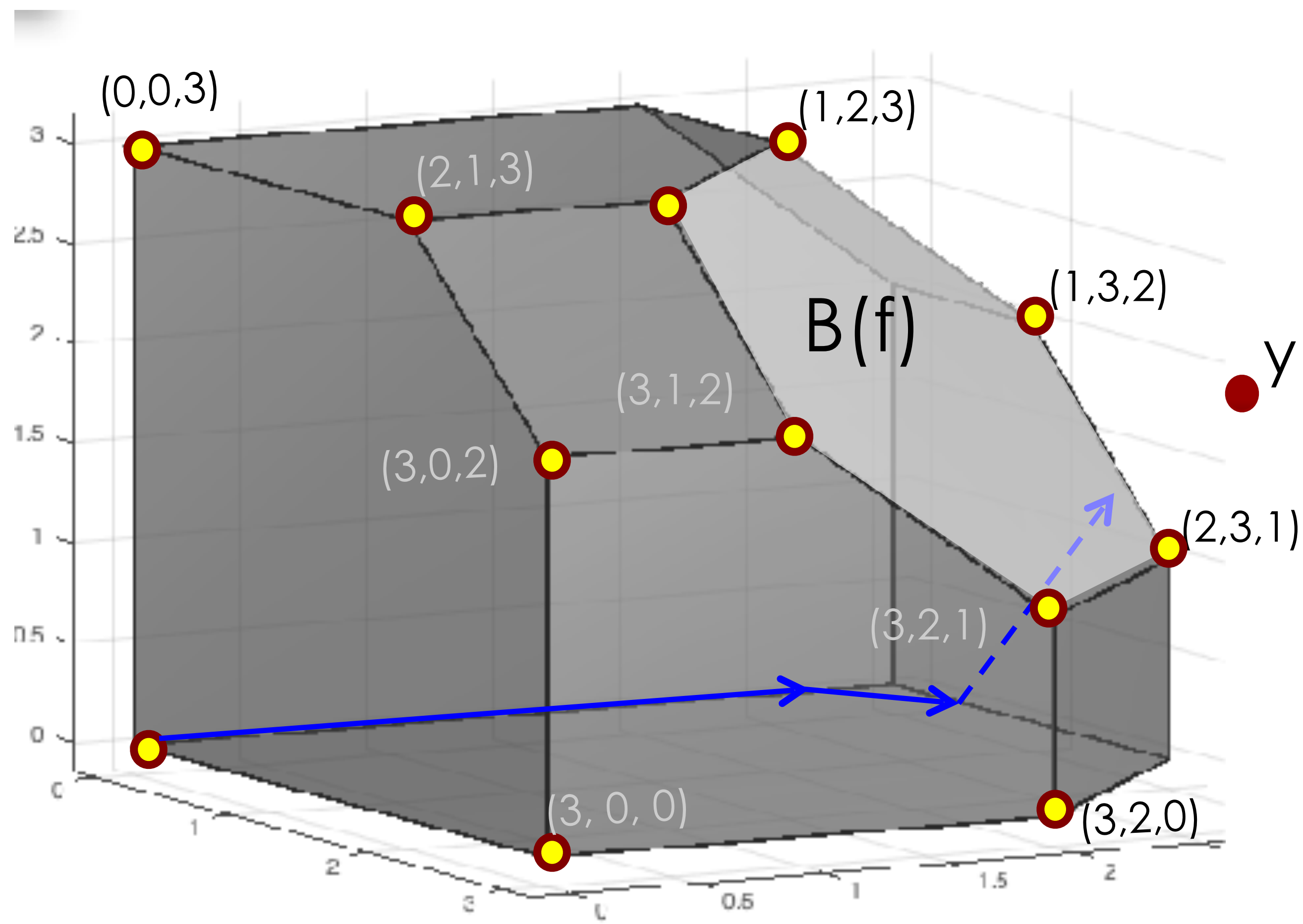
[Gupta, Goemans, Jaillet]

Inc-Fix Algorithm

Project: **y = (1.4, 4, 1)^T**
under Euclidean distance

$$\min_{x \in B(f)} \frac{1}{2} \sum_e (x(e) - y(e))^2$$





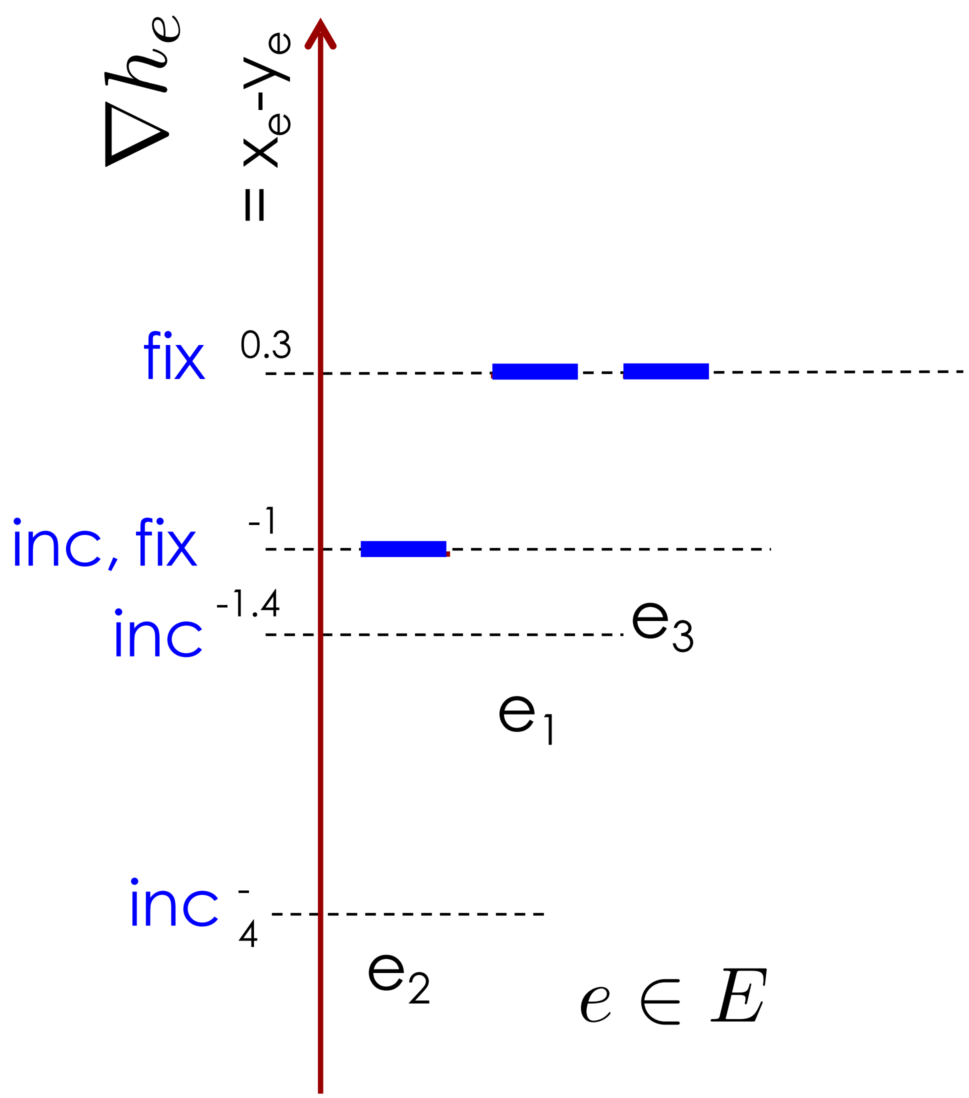
"greedy in gradient space" – proof from first-order optimality conditions

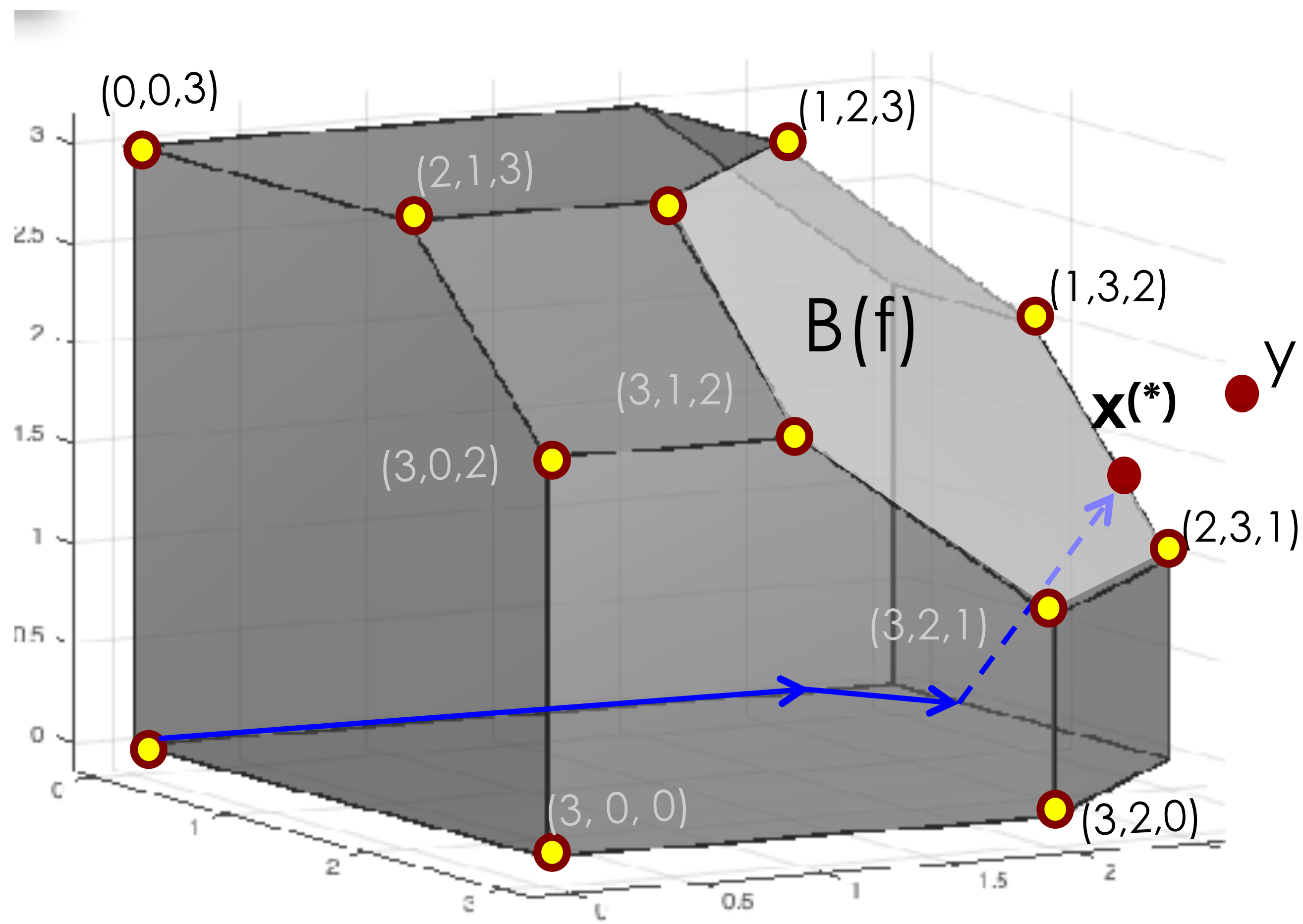
[Gupta, Goemans, Jaillet]

Inc-Fix Algorithm

Project: **$y = (1.4, 4, 1)^T$**
under Euclidean distance

$$\min_{x \in B(f)} \frac{1}{2} \sum_e (x(e) - y(e))^2$$





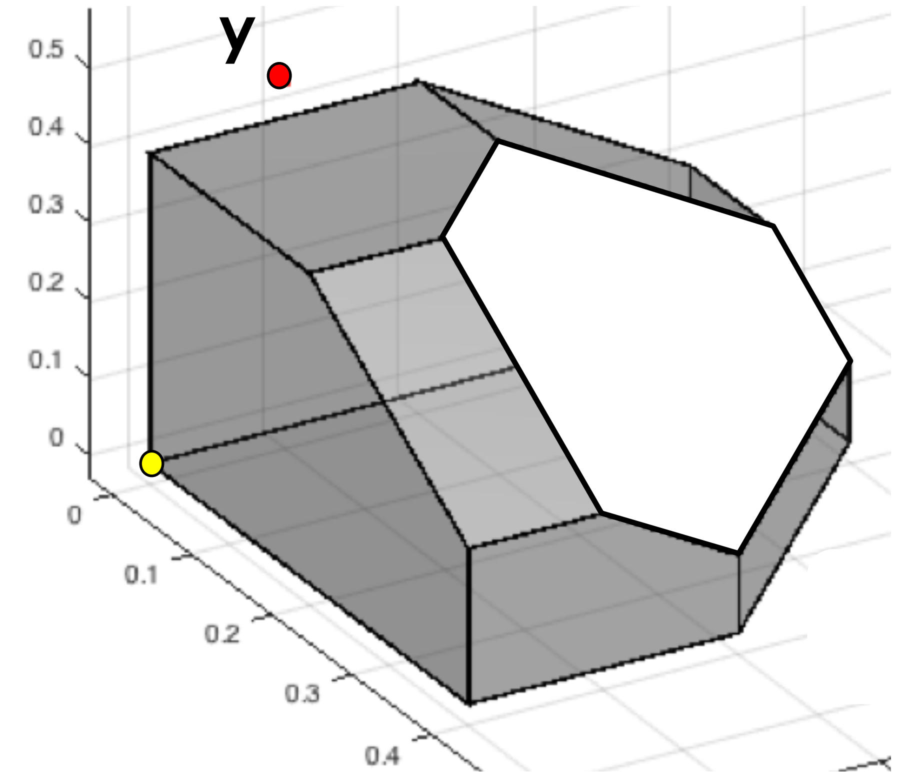
"greedy in gradient space" – proof from first-order optimality conditions

[Gupta, Goemans, Jaillet]

Running time

Squared Euclidean Distance, KL-Divergence:
Movement along lines

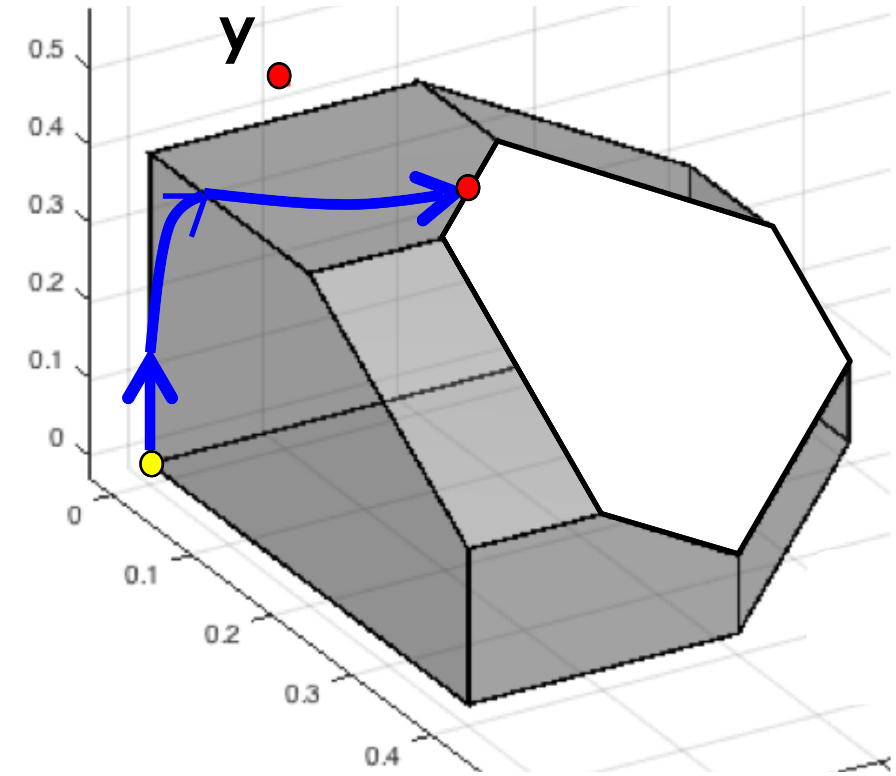
In general:
Piecewise smooth movement



Running time

Squared Euclidean Distance, KL-Divergence:
Movement along lines

In general:
Piecewise smooth movement

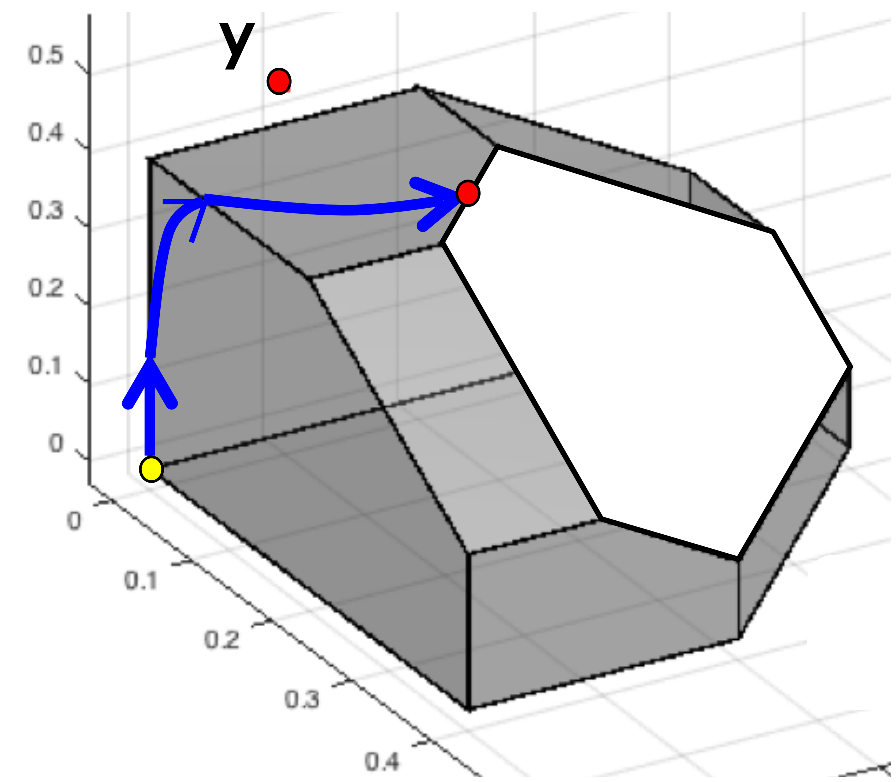


Running time

Squared Euclidean Distance, KL-Divergence:
Movement along lines

In general:
Piecewise smooth movement

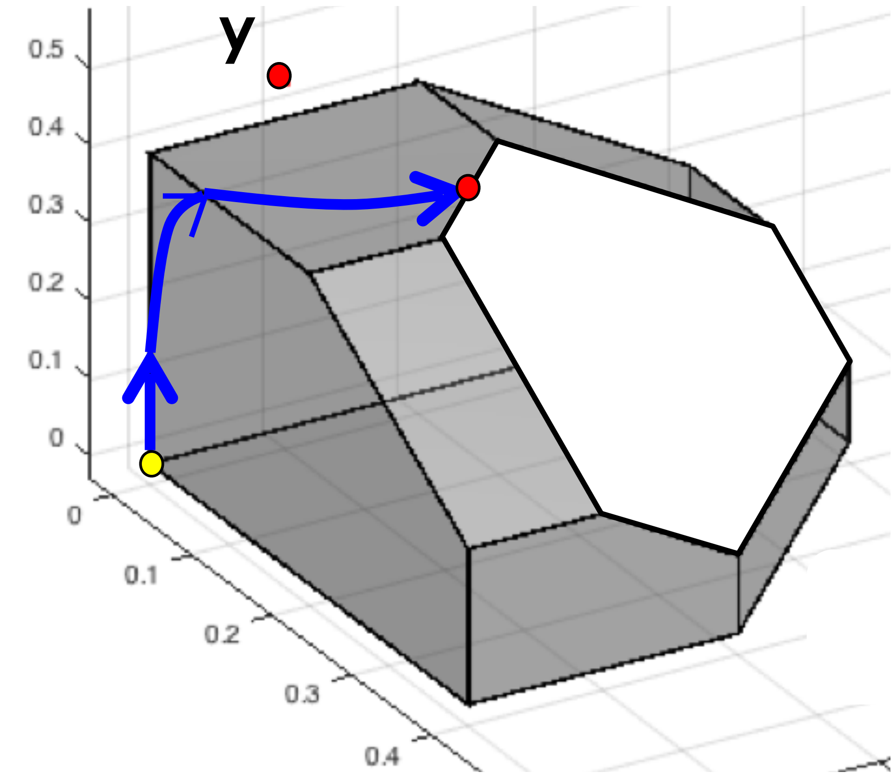
Details: How to do this movement?



Running time

Squared Euclidean Distance, KL-Divergence:
Movement along lines

In general:
Piecewise smooth movement



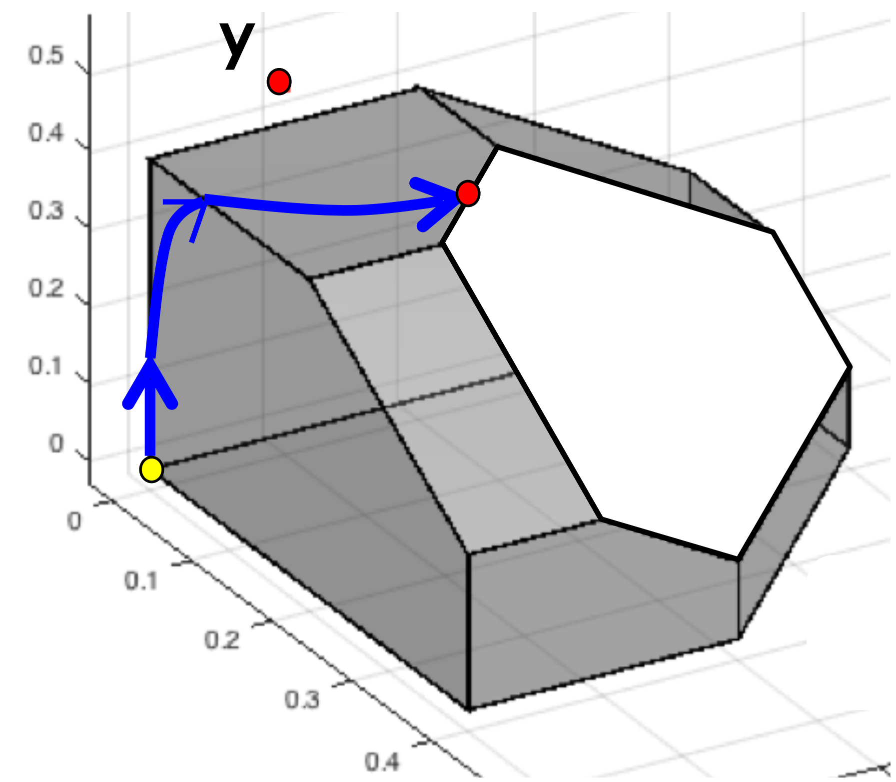
Details: How to do this movement?

O(n) Line Searches + non-linear equations in a single variable

Running time

Squared Euclidean Distance, KL-Divergence:
Movement along lines

In general:
Piecewise smooth movement



Details: How to do this movement?

O(n) Line Searches + non-linear equations in a single variable

Running time?

Using structural properties, we show Inc-Fix can be implemented in, in general,

O(n) Submodular Function Minimizations*

LSW'15: $O(n^4 \log^{O(1)} n + \gamma n^3 \log^2 n)$

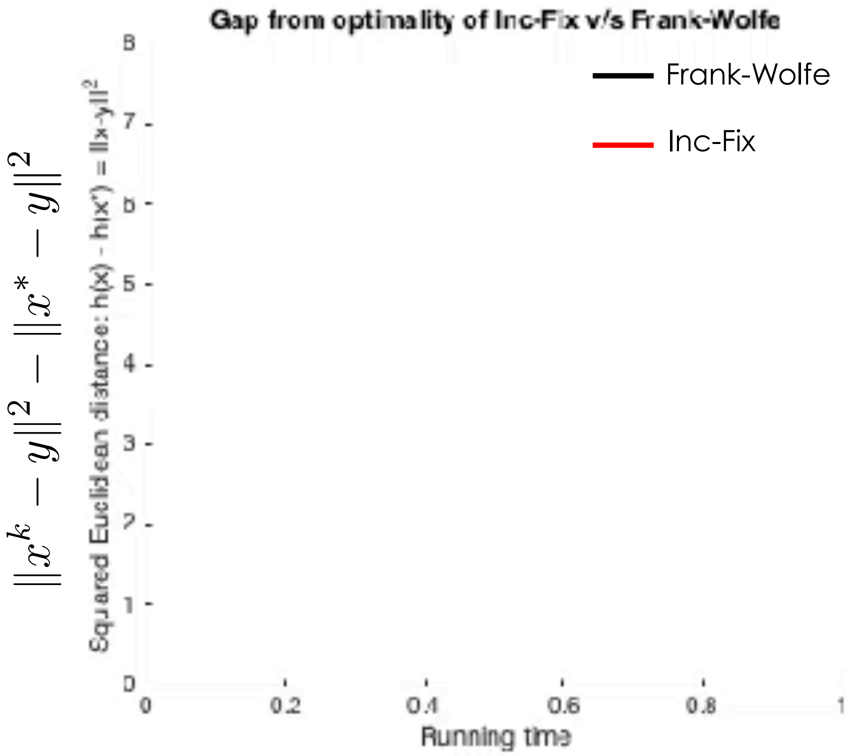
$O(n^3 \log^{O(1)}(nM) + \gamma n^2 \log(nM))$

CLSW'16: $O(\gamma n M^3 \log n)$

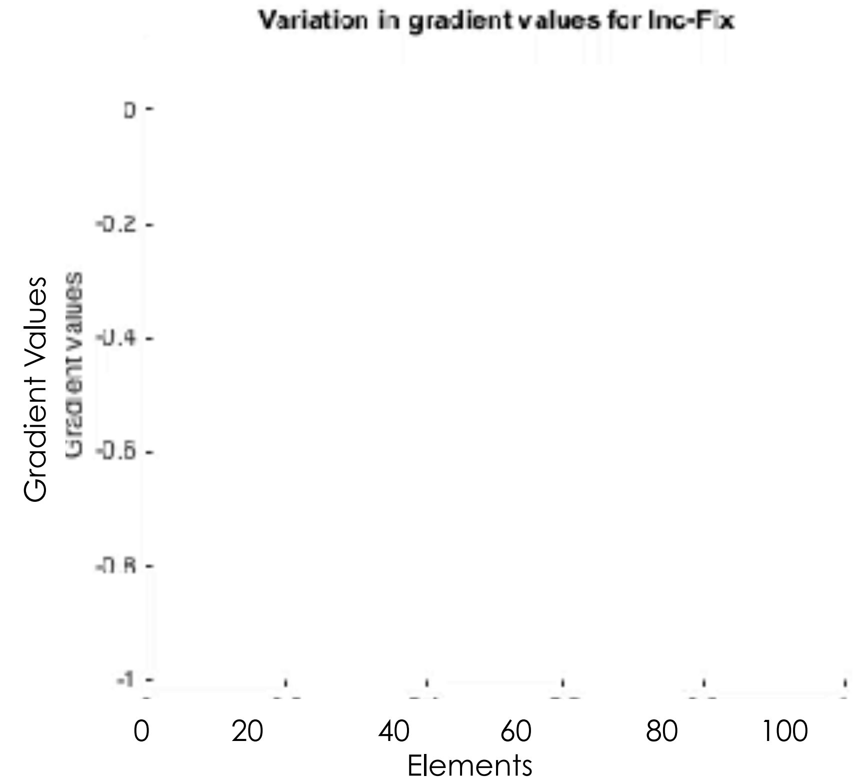
*Require maximal minimizers, note that checking for feasibility itself requires a SFM.

Computations for cardinality-based f(.)

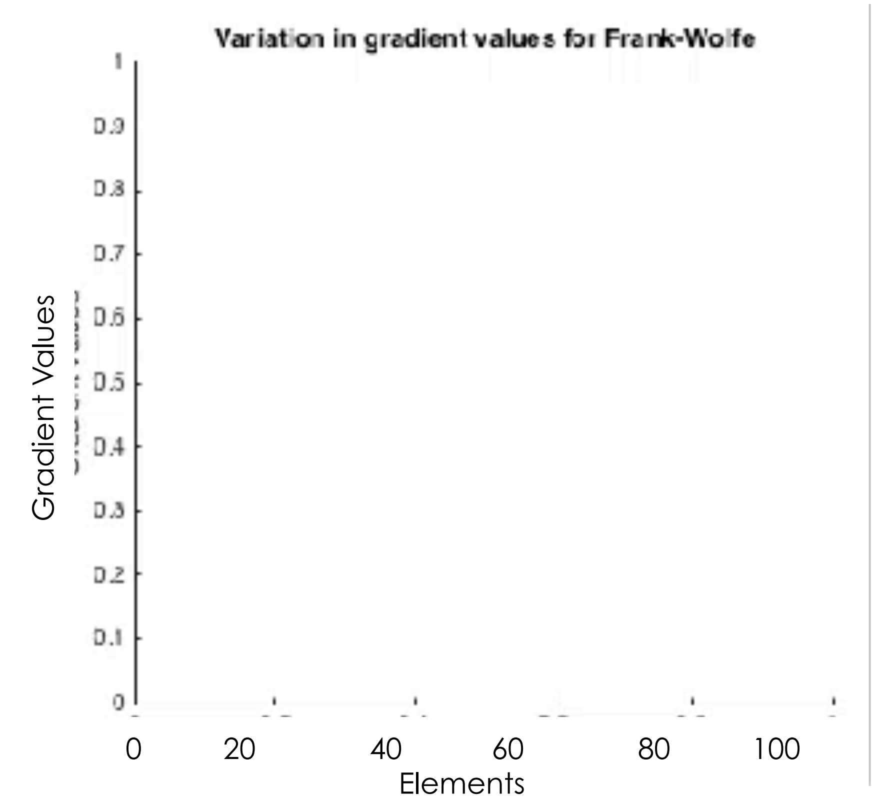
Gap from optimality



Inc-Fix



Frank-Wolfe

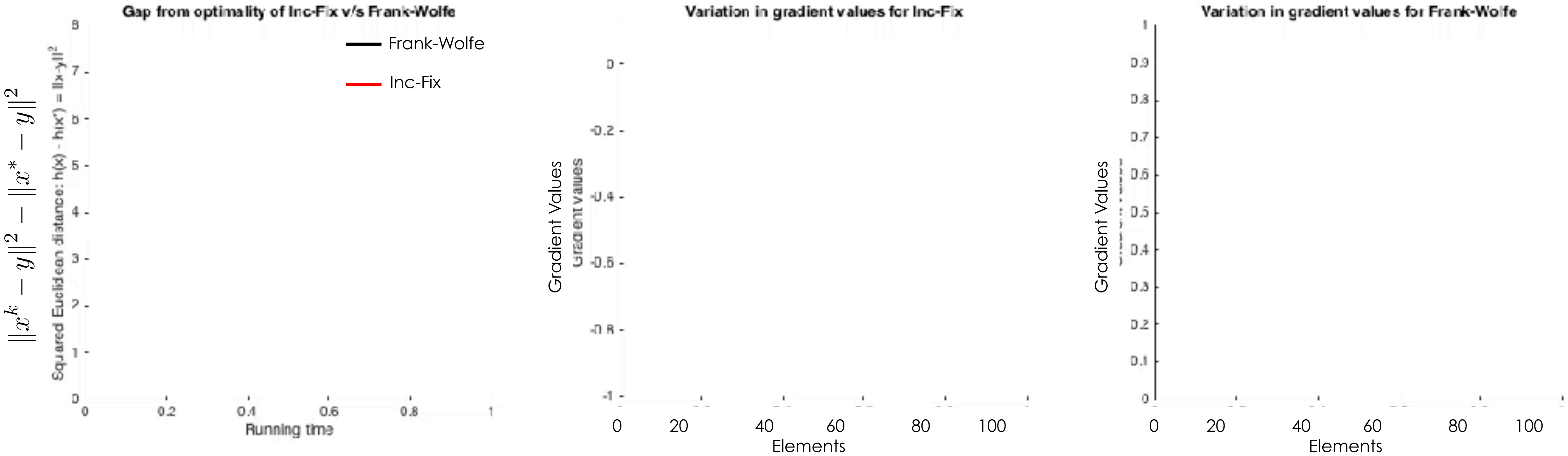


For cardinality-based functions, Inc-Fix takes $O(n^2)$ for exact, while vanilla FW takes $O(n \ln n * C_h/\epsilon)$ for ϵ-approx.

(O(n (log n + k)) for simplex, k-subsets, k-truncated-permutations)

Computations for cardinality-based f(.)

Gap from optimality | Inc-Fix | Frank-Wolfe

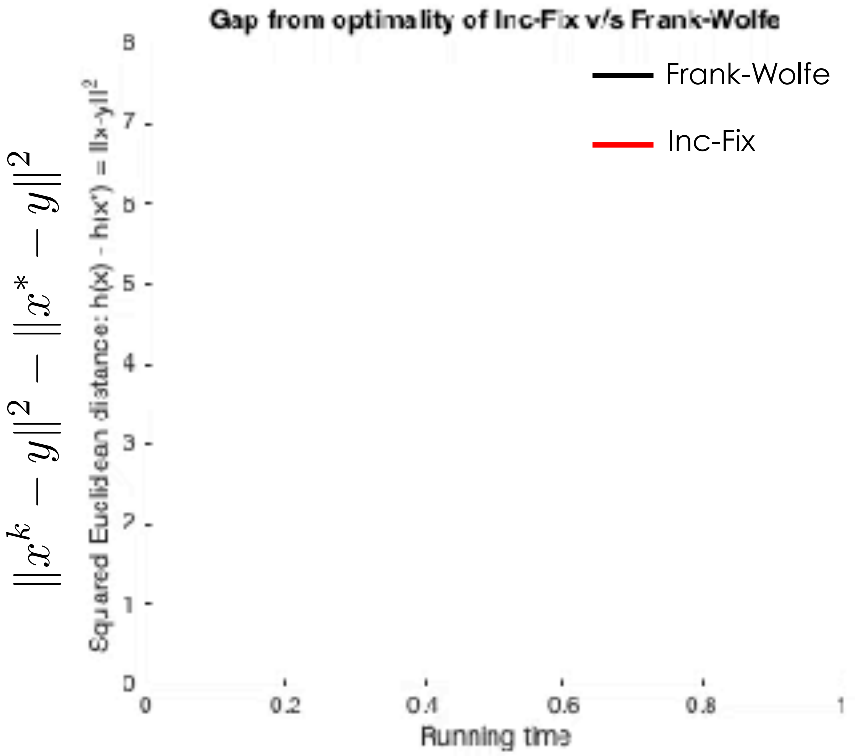


For cardinality-based functions, Inc-Fix takes $O(n^2)$ for exact, while vanilla FW takes $O(n \ln n * C_h/\epsilon)$ for ϵ-approx.

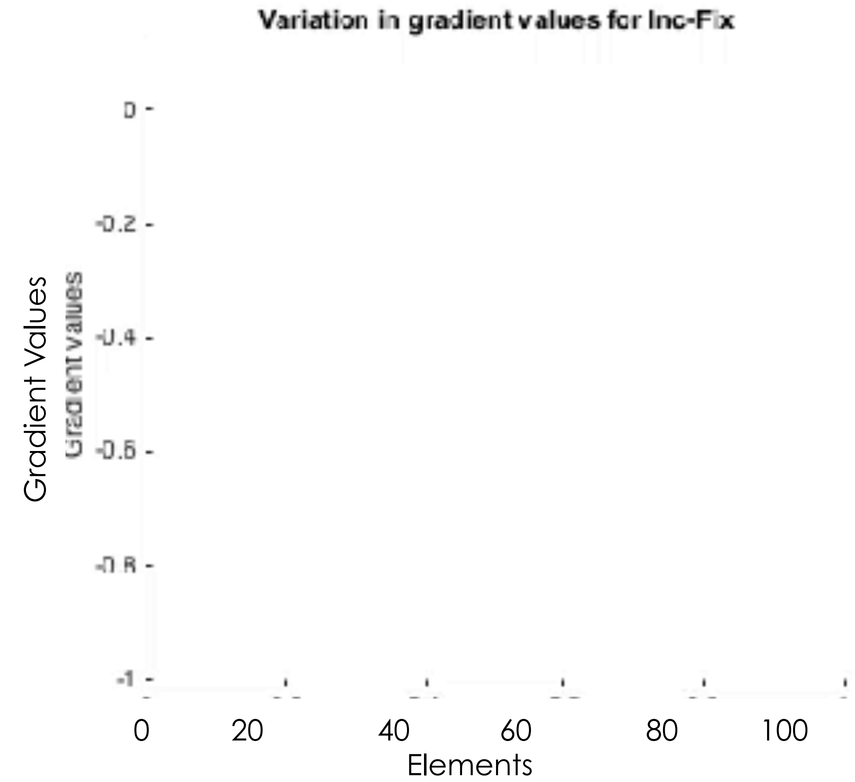
(O(n (log n + k)) for simplex, k-subsets, k-truncated-permutations)

Computations for cardinality-based f(.)

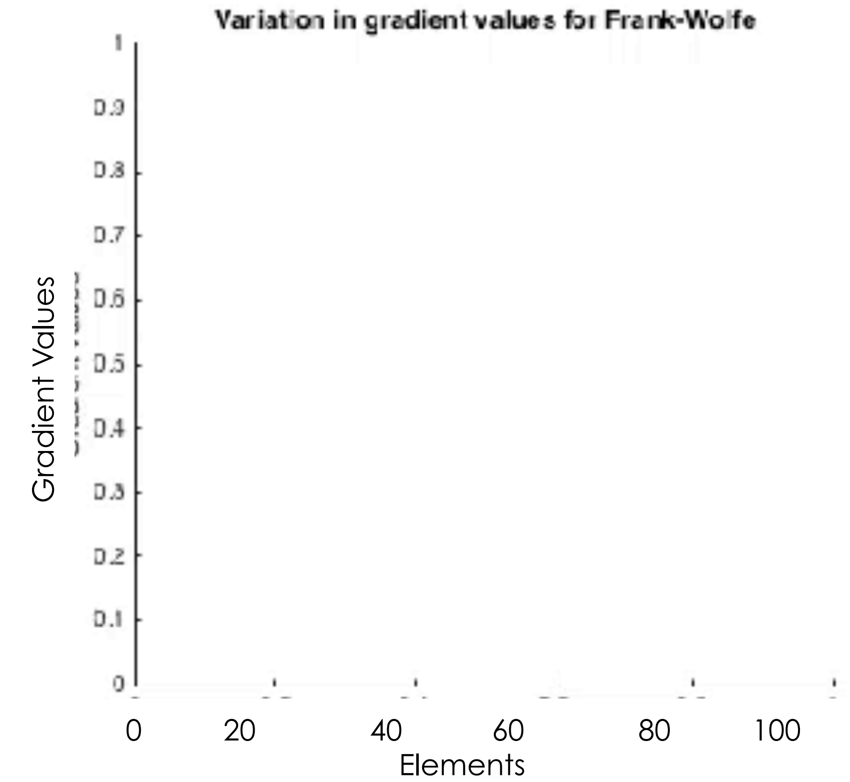
Gap from optimality



Inc-Fix



Frank-Wolfe



For cardinality-based functions, Inc-Fix takes $O(n^2)$ for exact, while vanilla FW takes $O(n \ln n * C_h/\epsilon)$ for ϵ-approx.

(O(n (log n + k)) for simplex, k-subsets, k-truncated-permutations)

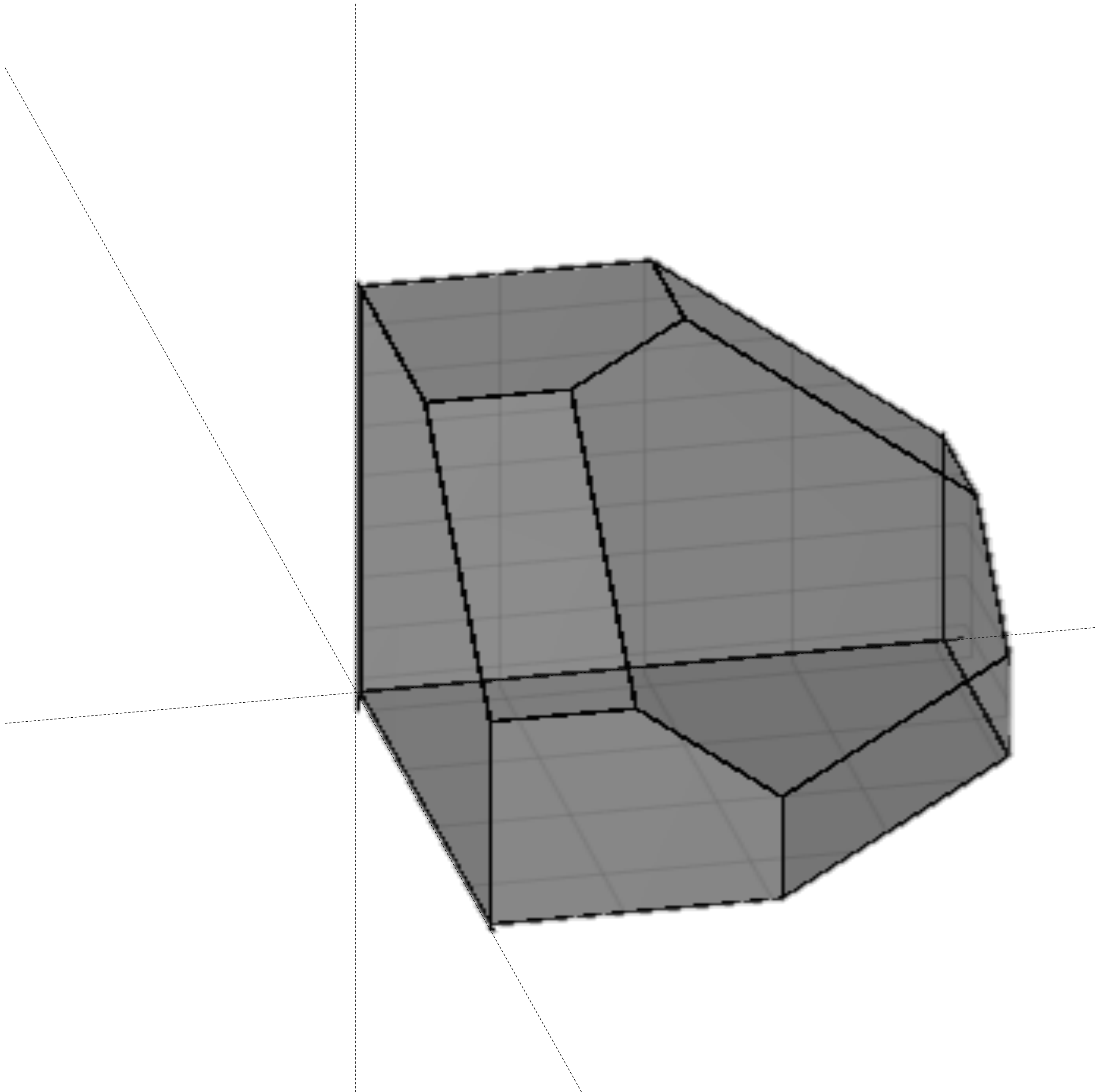
Outline

1. **Projections**
   - Motivation
   - Problem setup
   - **Novel algorithm**: Inc-Fix for separable convex minimization:
     - Main Result: O(n) SFM or O(n) Line searches
     - Exact computations, modulo solving a univariate equation

2. **Line Searches**
   - Previous best known: Megiddo's parametric search
   - Using Newton's Discrete Method: $n^2 + n \log^2 n$ SFM (n^6 improvement)

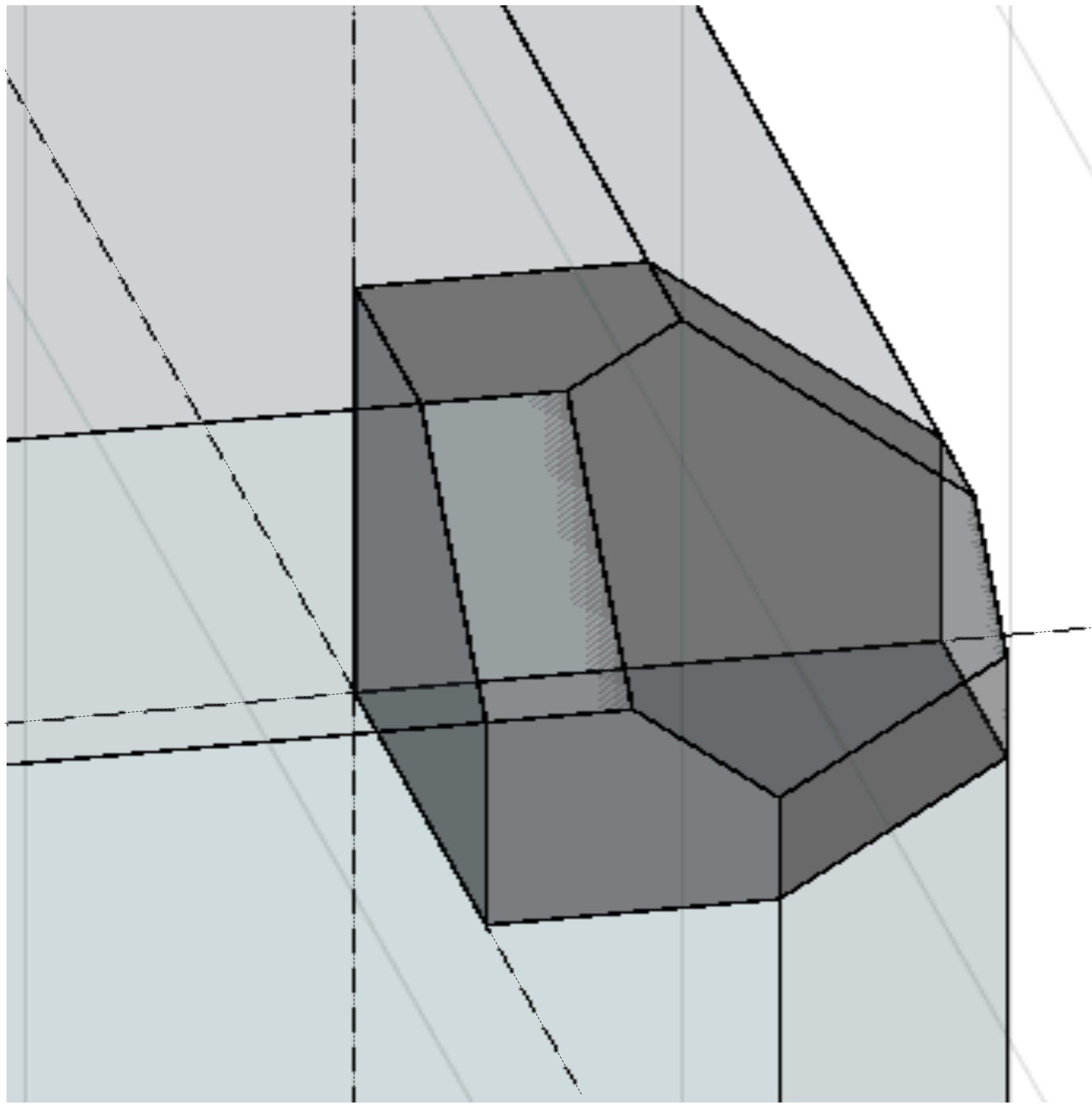
3. **What works best when**
   - Problems with Max-Cut and QUBO heuristics comparative studies
   - **Our framework**: Expanded instance library, Implementation of 37 heuristics, Large-scale cloud computing on the cross product
   - **Hyper-heuristic**: Map every instance to a feature space, learn "performance" of heuristics

3. Feasibility along a Line



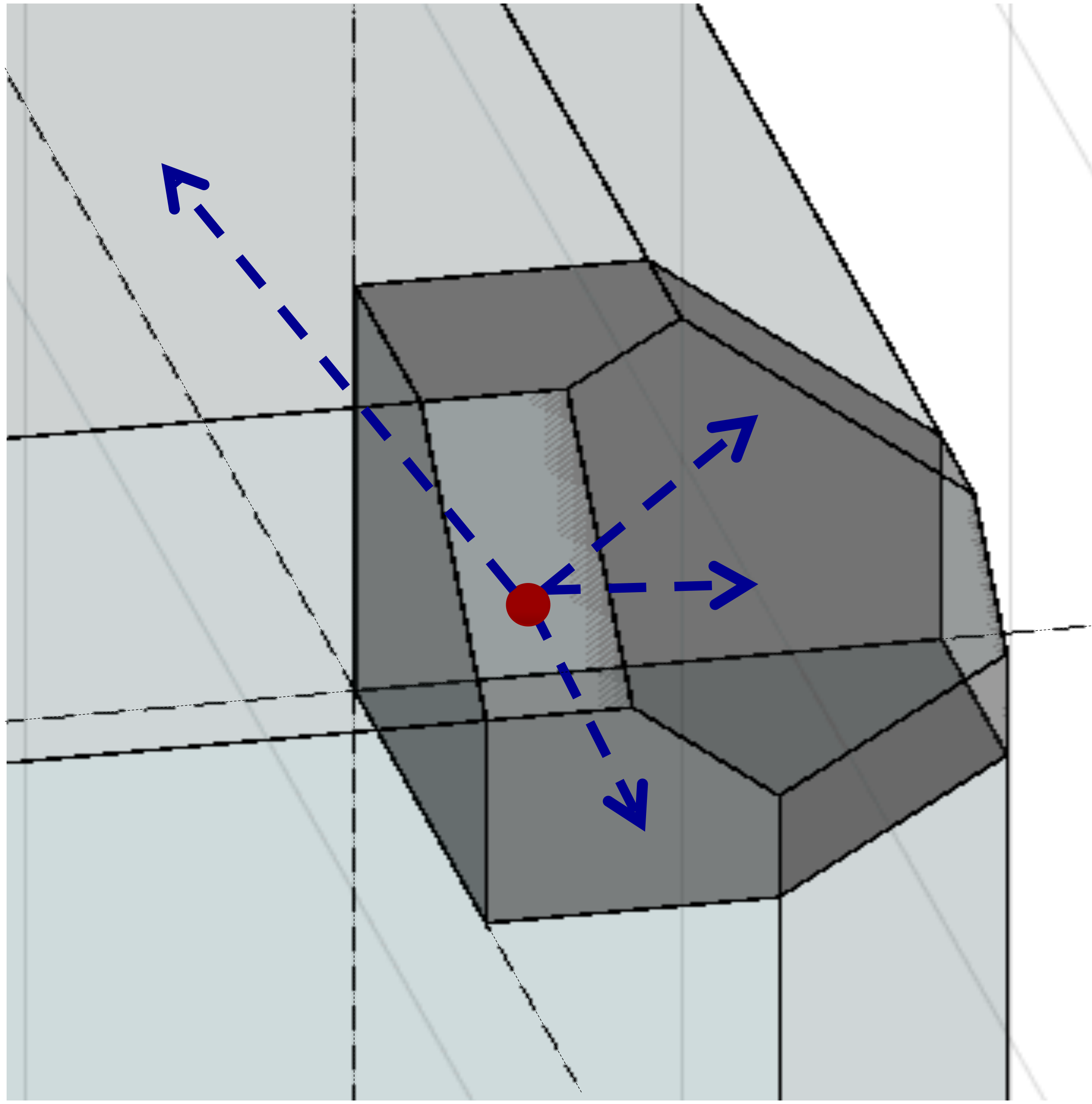
How much to move in a direction while staying feasible?

3. Feasibility along a Line



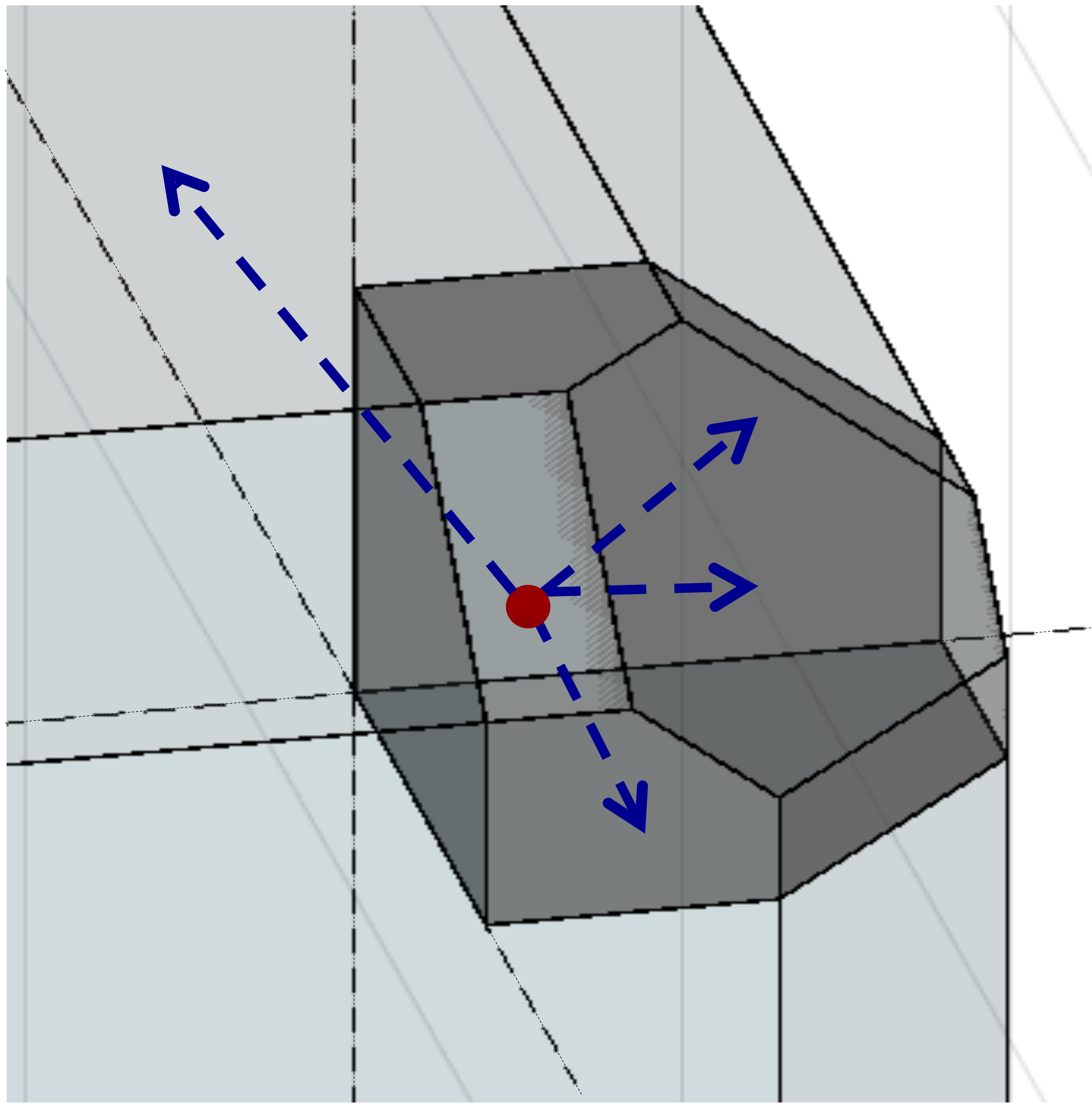
How much to move in a direction while staying feasible?

3. Feasibility along a Line



How much to move in a direction while staying feasible?

3. Feasibility along a Line

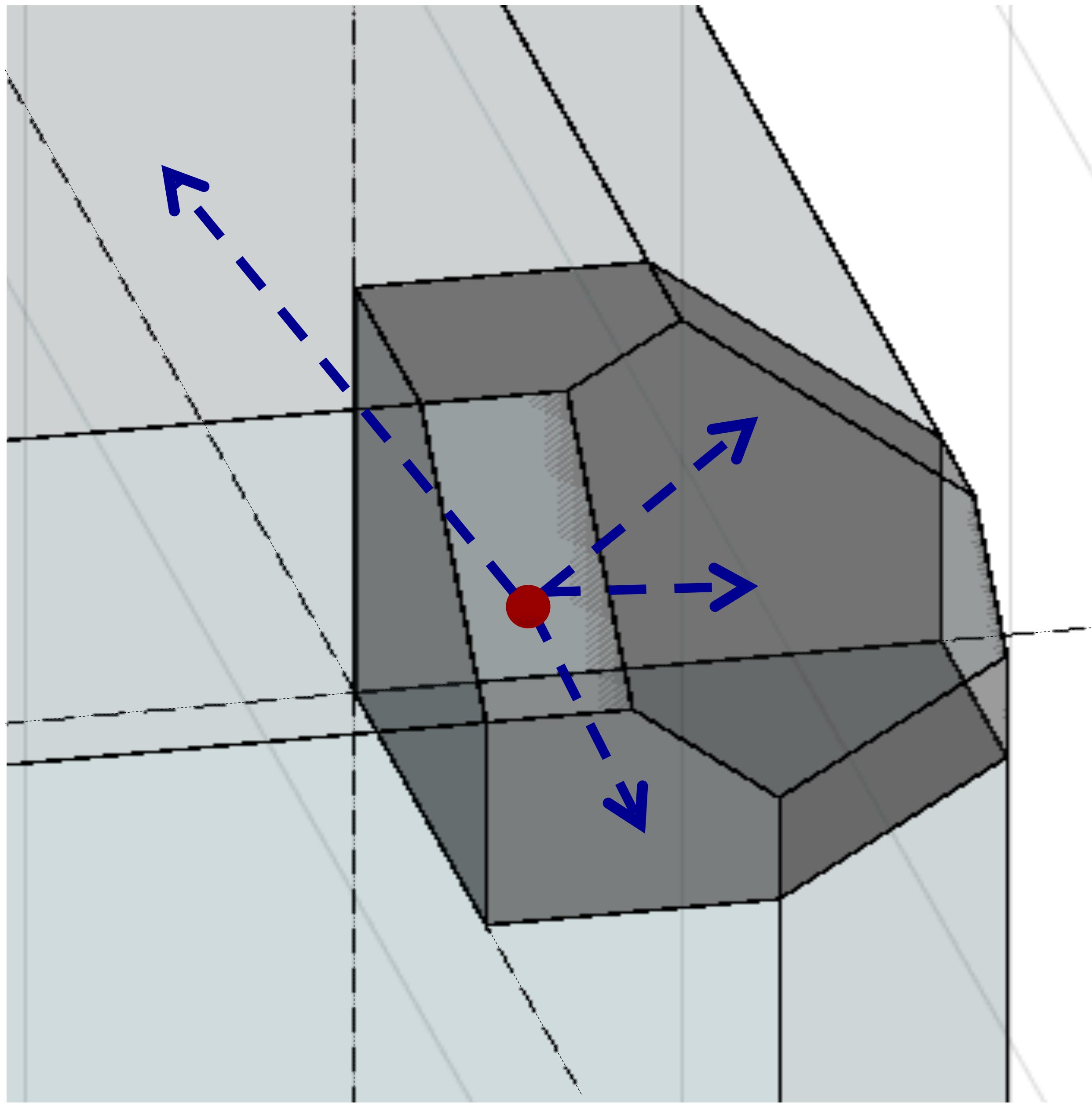


Sub-problem in many methods:

- Inc-Fix, of course
- Frank-Wolfe
  [Frank, Wolfe, Jaggi, Lacoste-Julien, Freund, Grigas, …]
- Caratheodory's Theorem

How much to move in a direction while staying feasible?

3. Feasibility along a Line



How much to move in a direction while staying feasible?

Sub-problem in many methods:

- Inc-Fix, of course
- Frank-Wolfe
  [Frank, Wolfe, Jaggi, Lacoste-Julien, Freund, Grigas, …]
- Caratheodory's Theorem

Others:

- Densest sub-graphs
  [Nagano et al. 2011]
- Minimum Ratio Problems
  [Cunningham 1985]

Line Search

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **$\tilde{O}(n^8)$ SFM** [Nagano 2011]

Line Search

$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n⁸) SFM** [Nagano 2011]

Line Search

$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$

Line Search

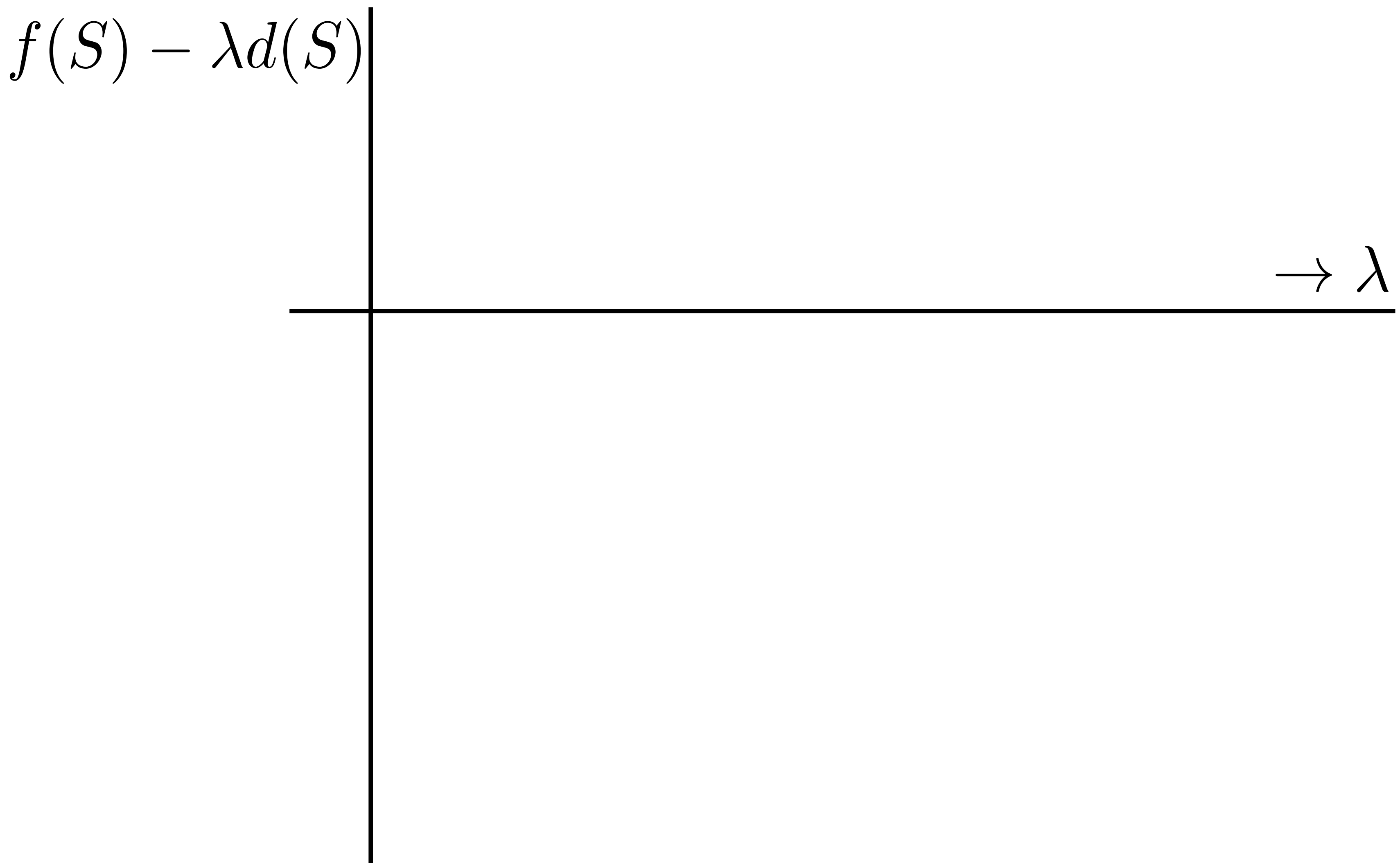
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n⁸) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

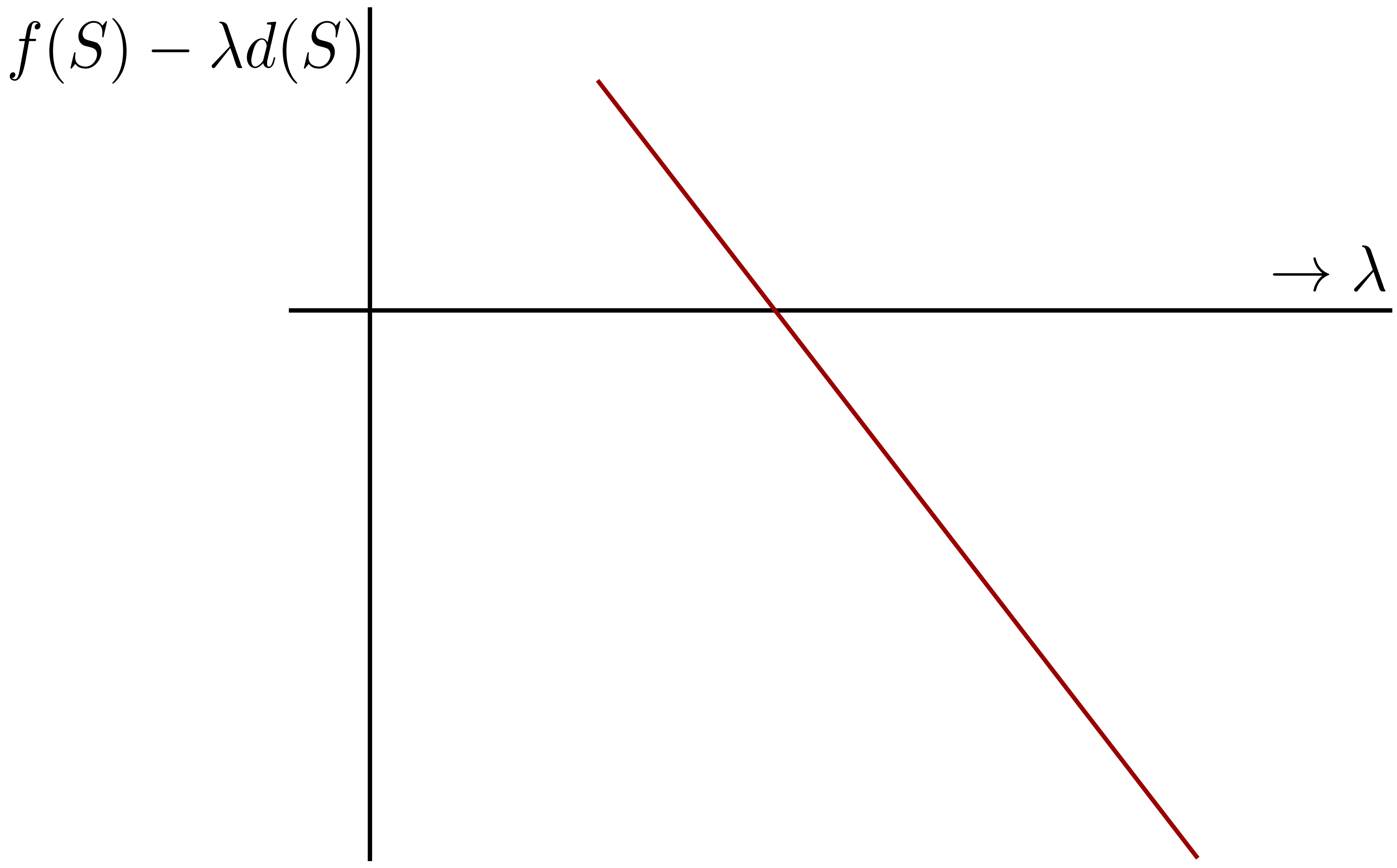
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

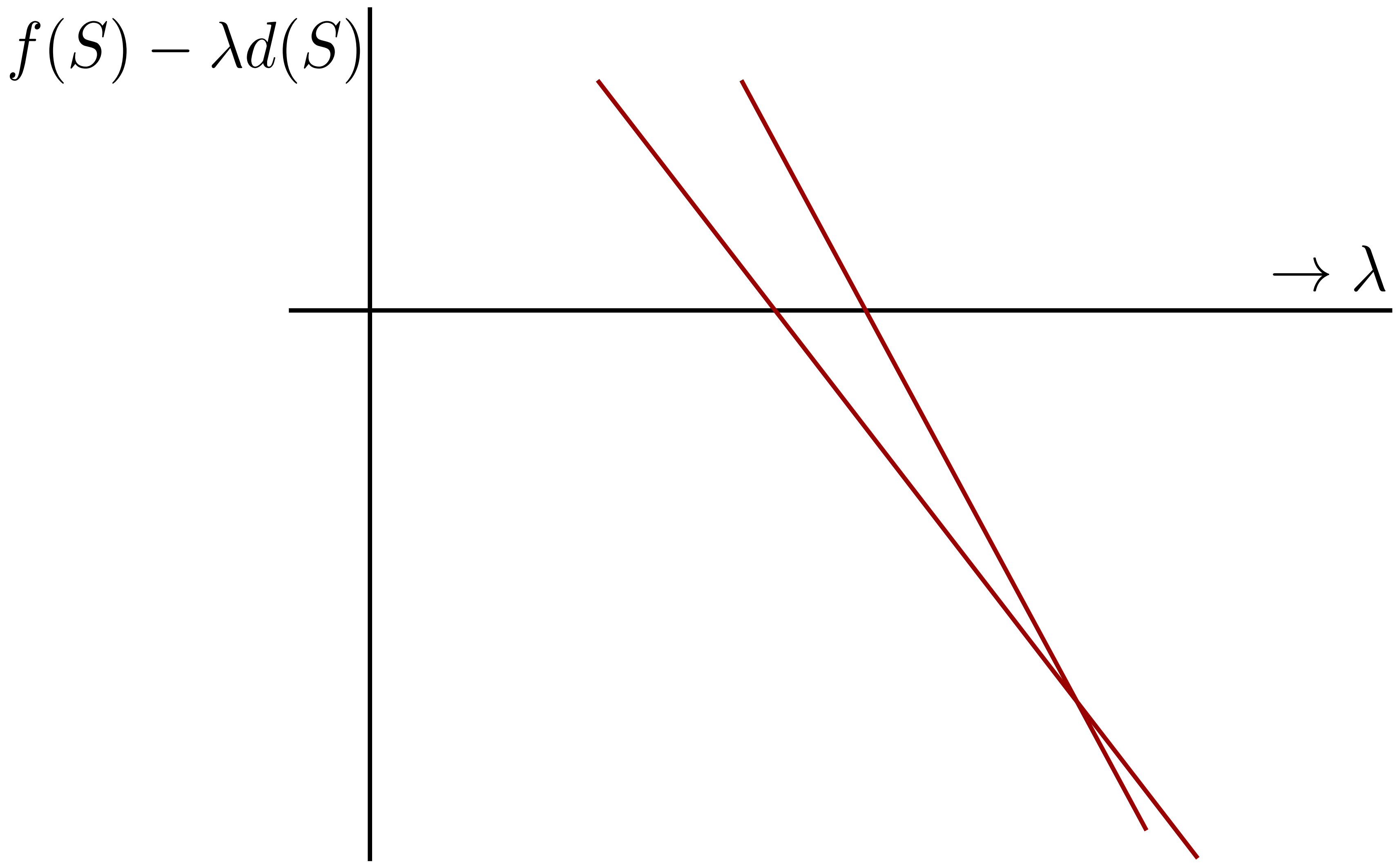
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

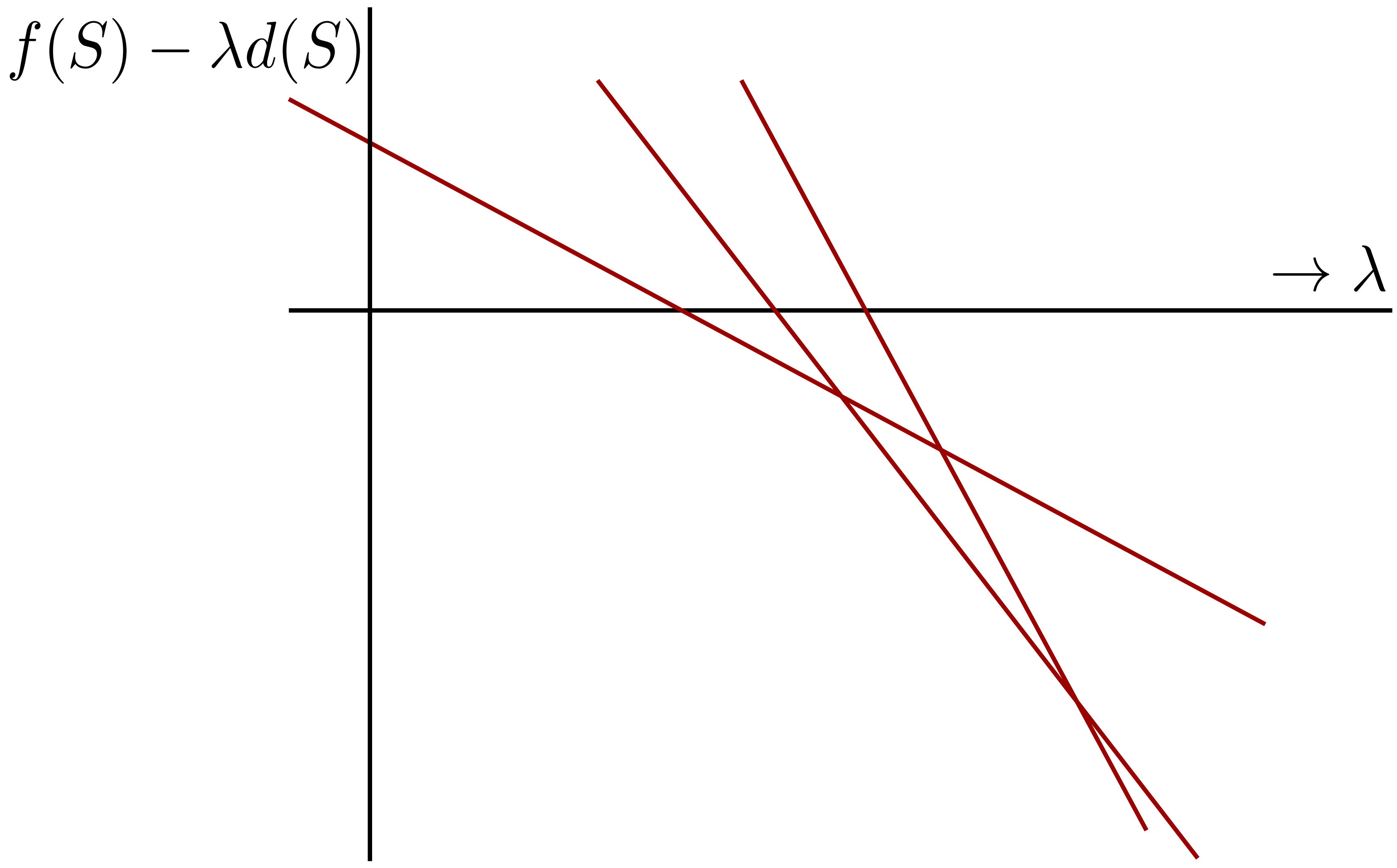
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

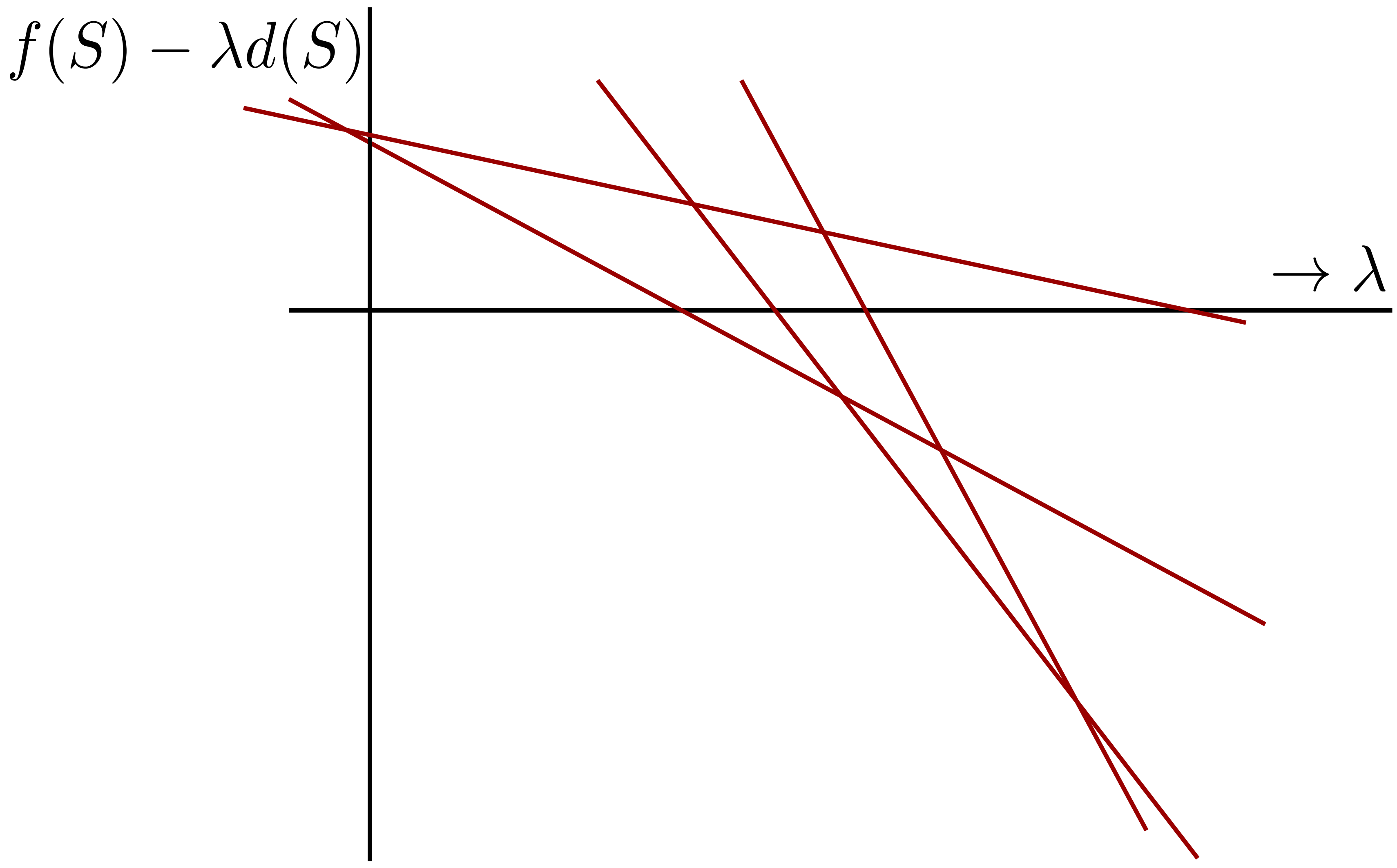
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

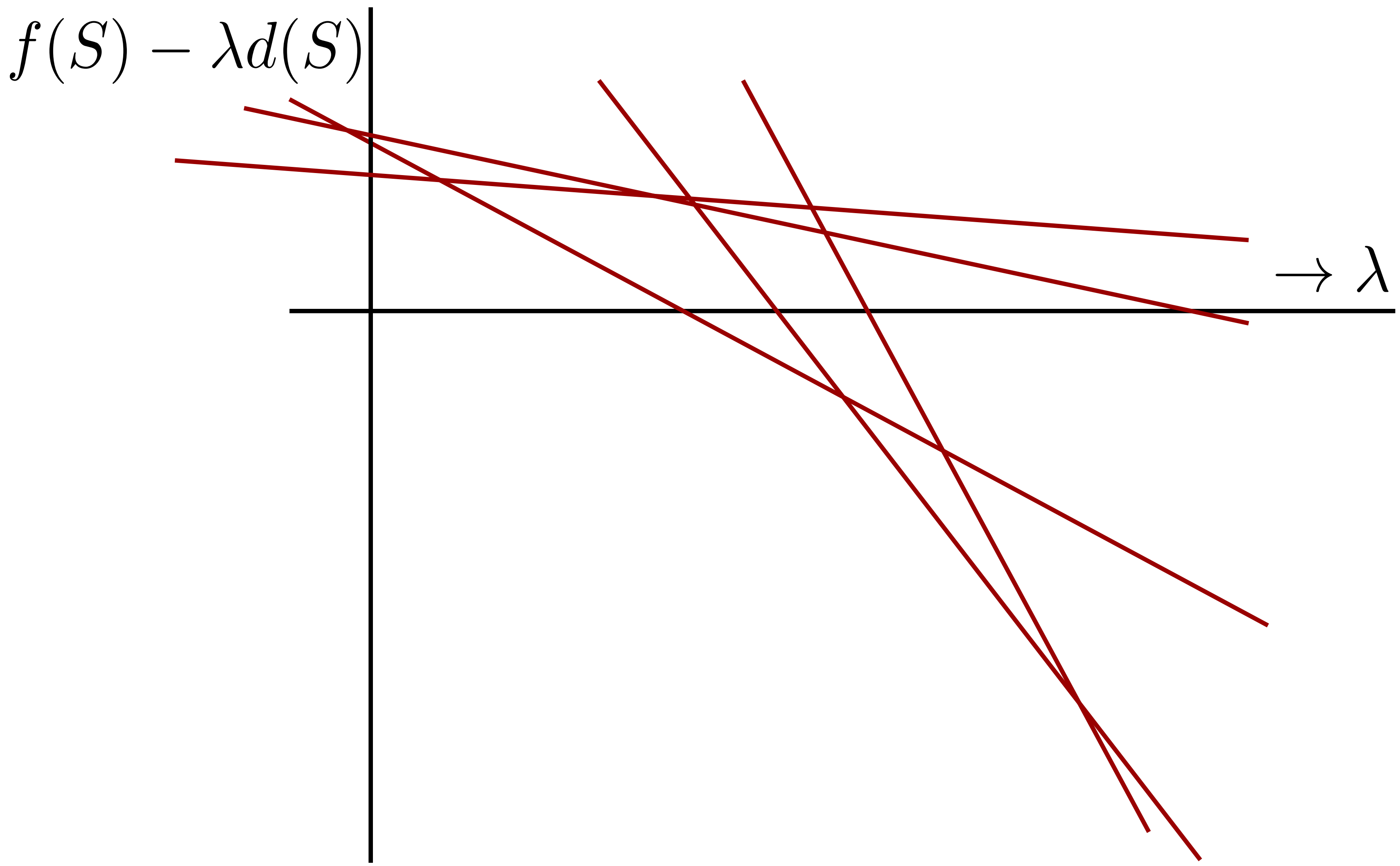
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

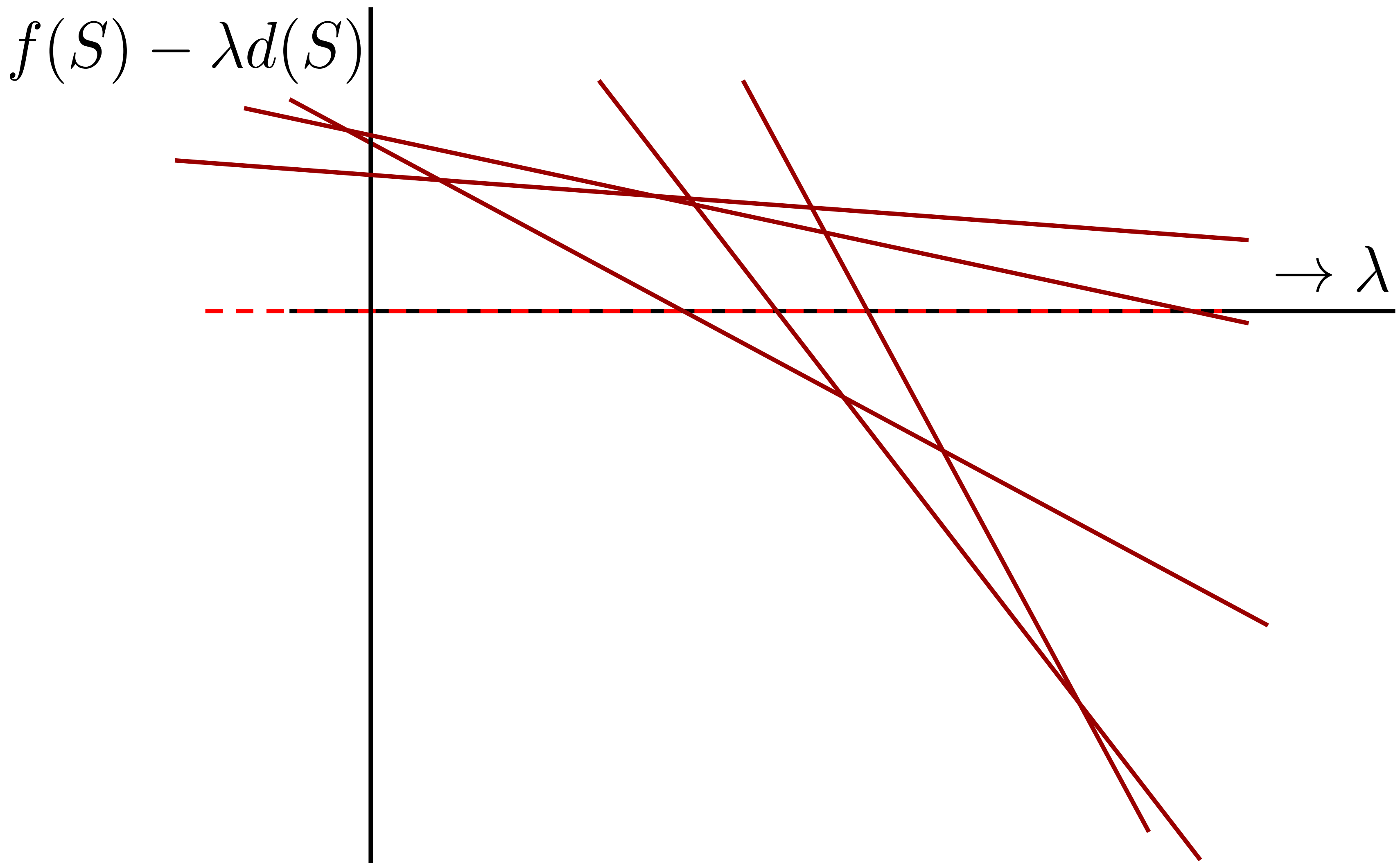
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

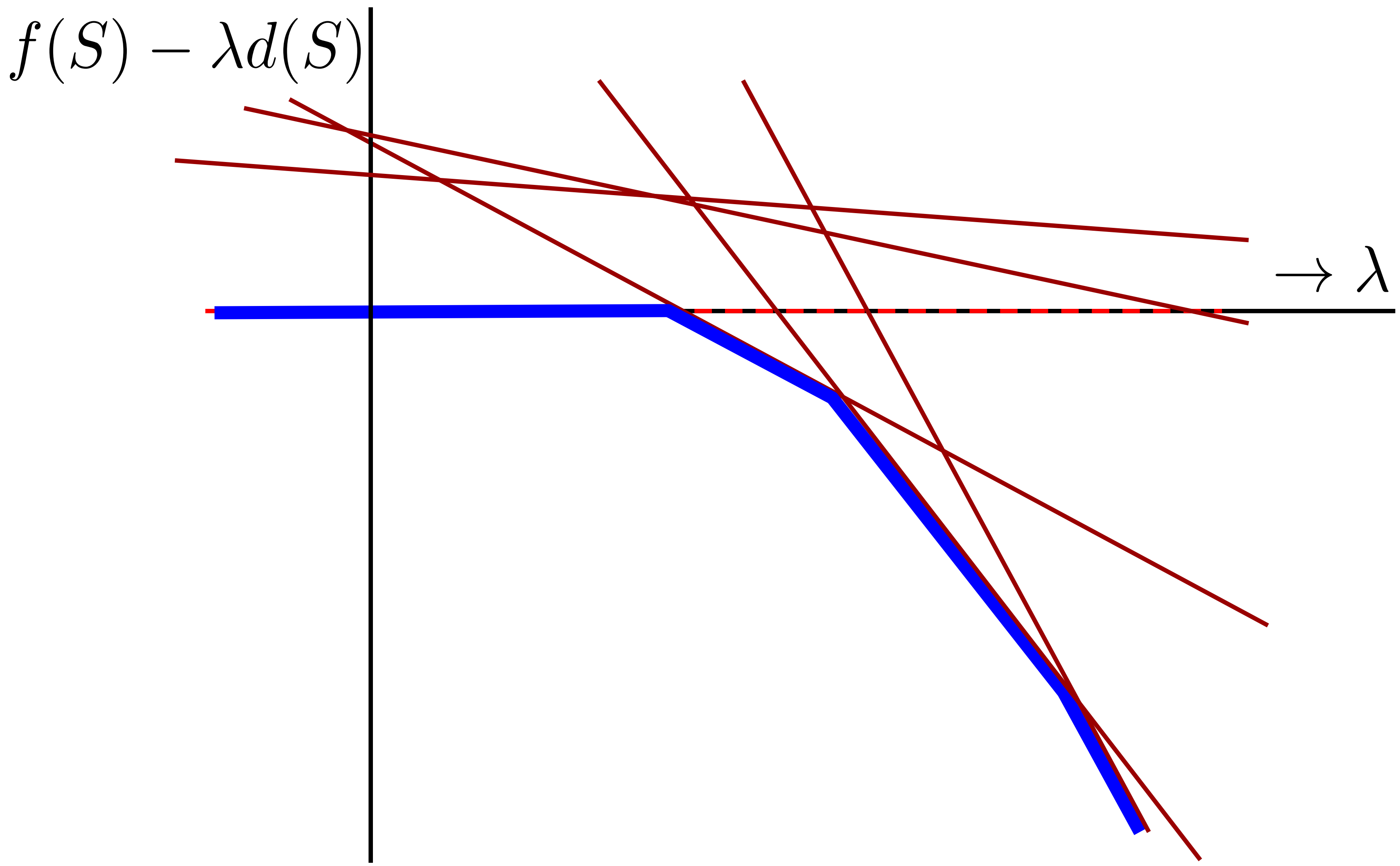
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n⁸) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

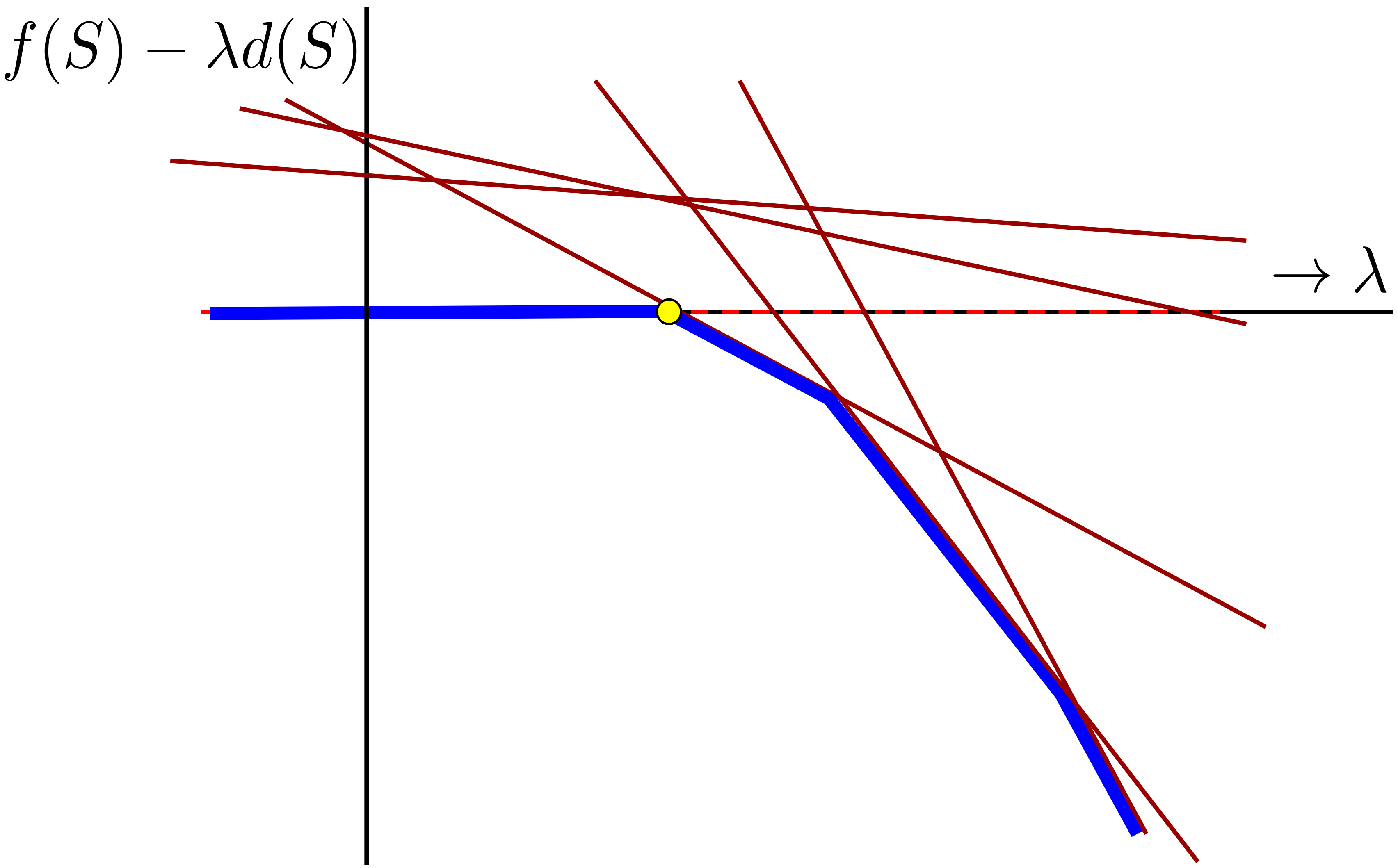
$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$



Line Search

$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

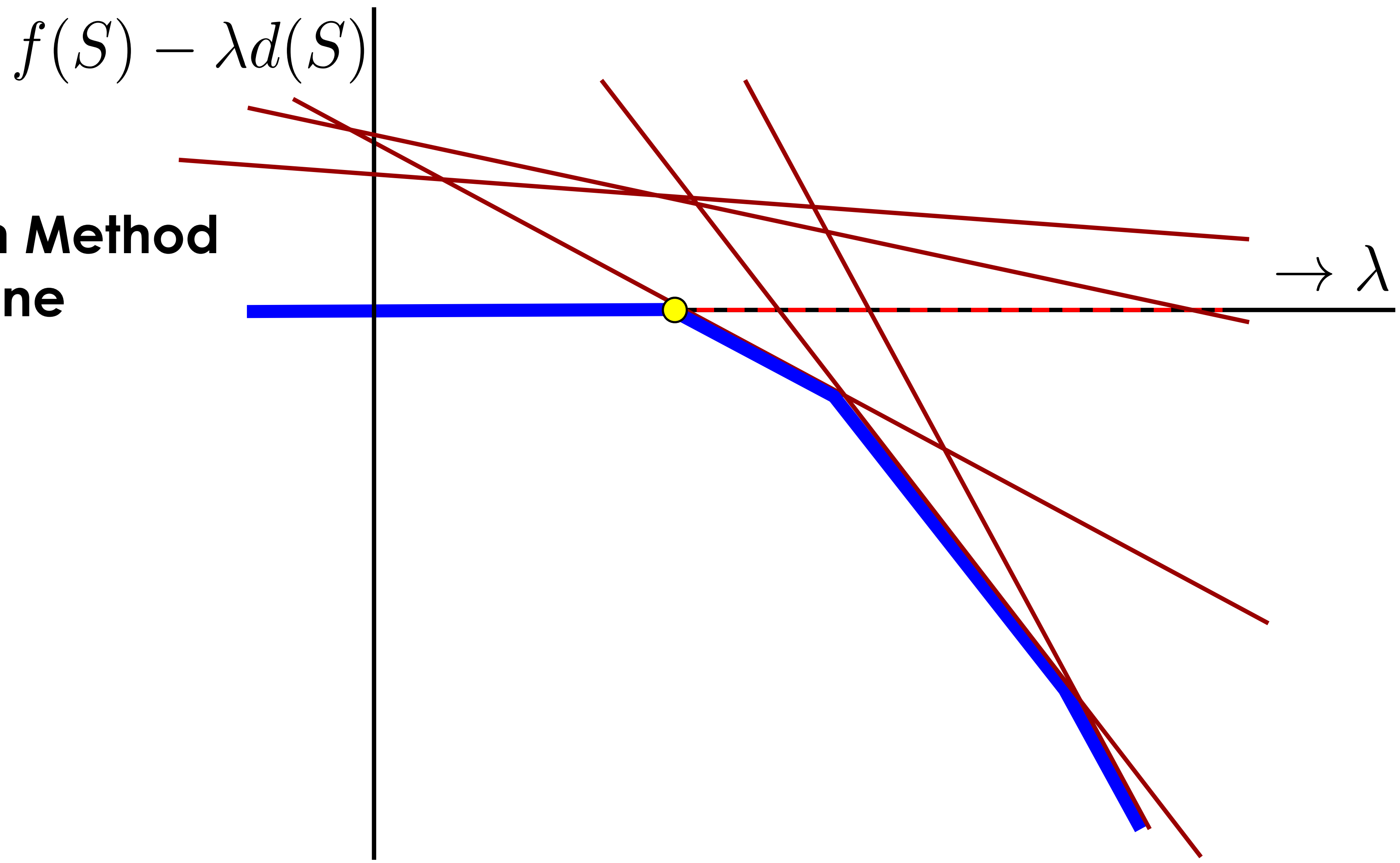
Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$

Discrete Newton Method for Parametric Line Search



Line Search

$$\sum_{e \in S} x(e) = x(S) \le f(S)$$

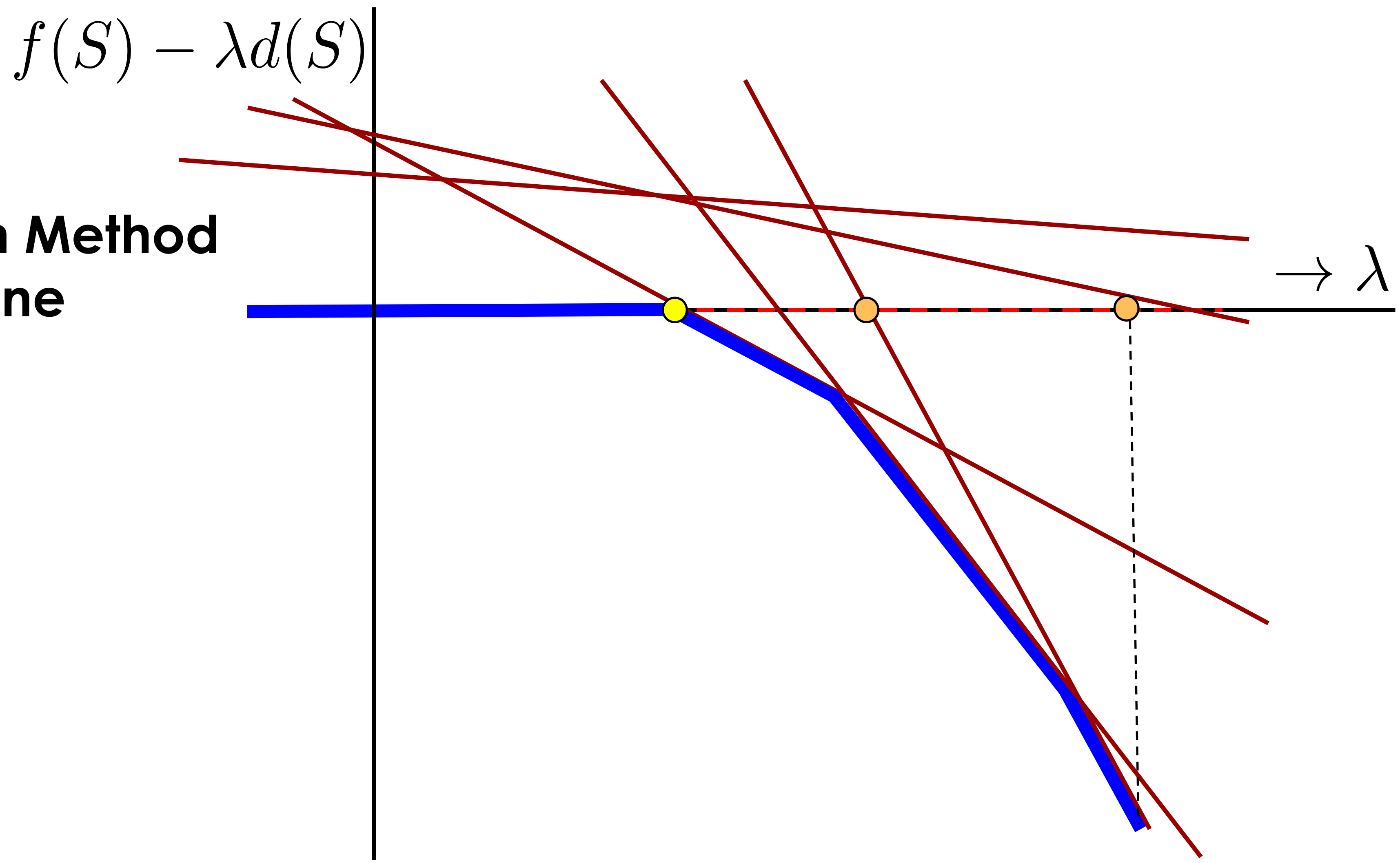
Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \ge 0$$

Discrete Newton Method for Parametric Line Search



Line Search

$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

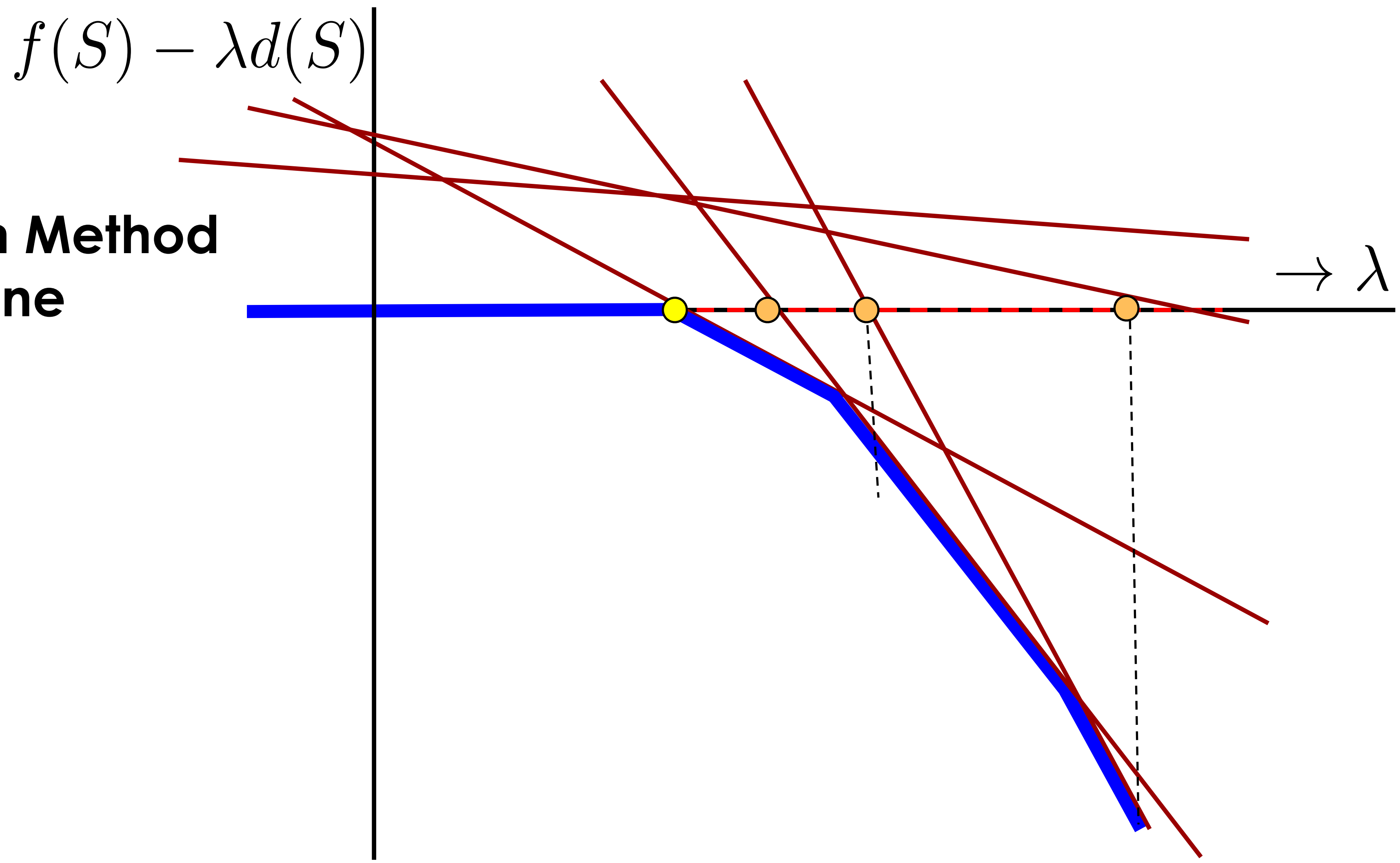
Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$

Discrete Newton Method for Parametric Line Search



Line Search

$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

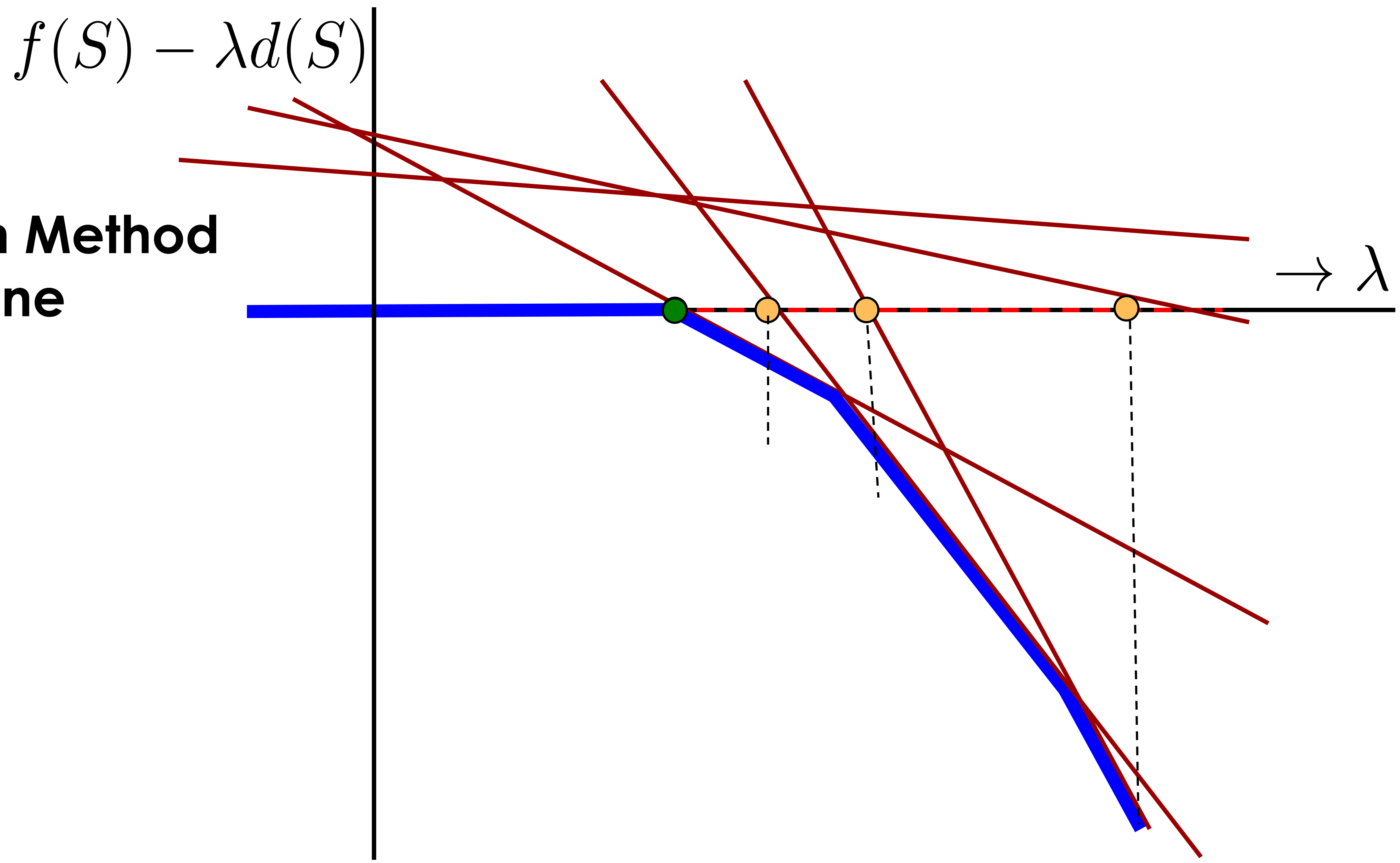
Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$

Discrete Newton Method for Parametric Line Search



Line Search

$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

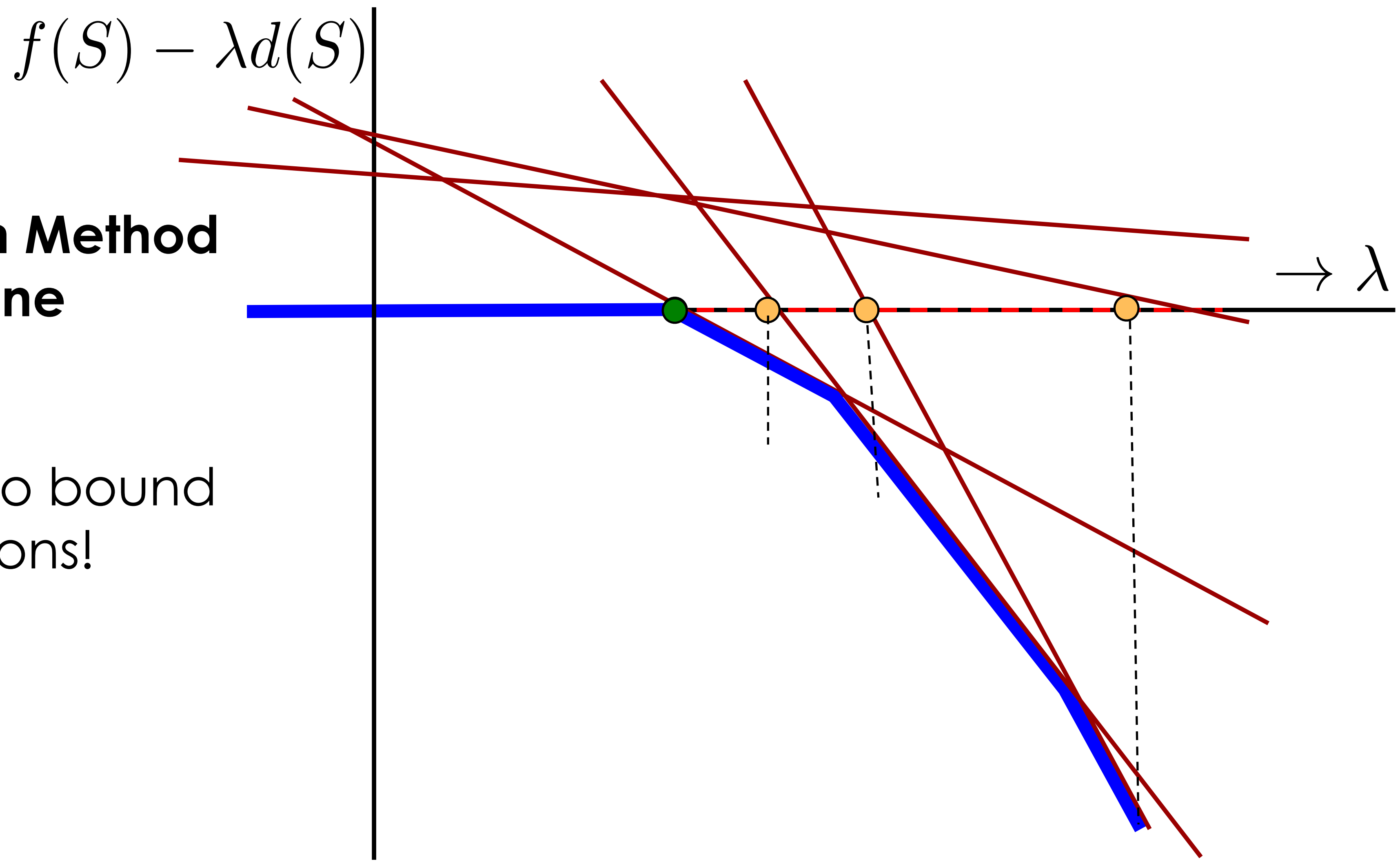
Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n⁸) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \Rightarrow \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$

Discrete Newton Method for Parametric Line Search



Open question to bound the no. of iterations!

Line Search

$$\sum_{e \in S} x(e) = x(S) \leq f(S)$$

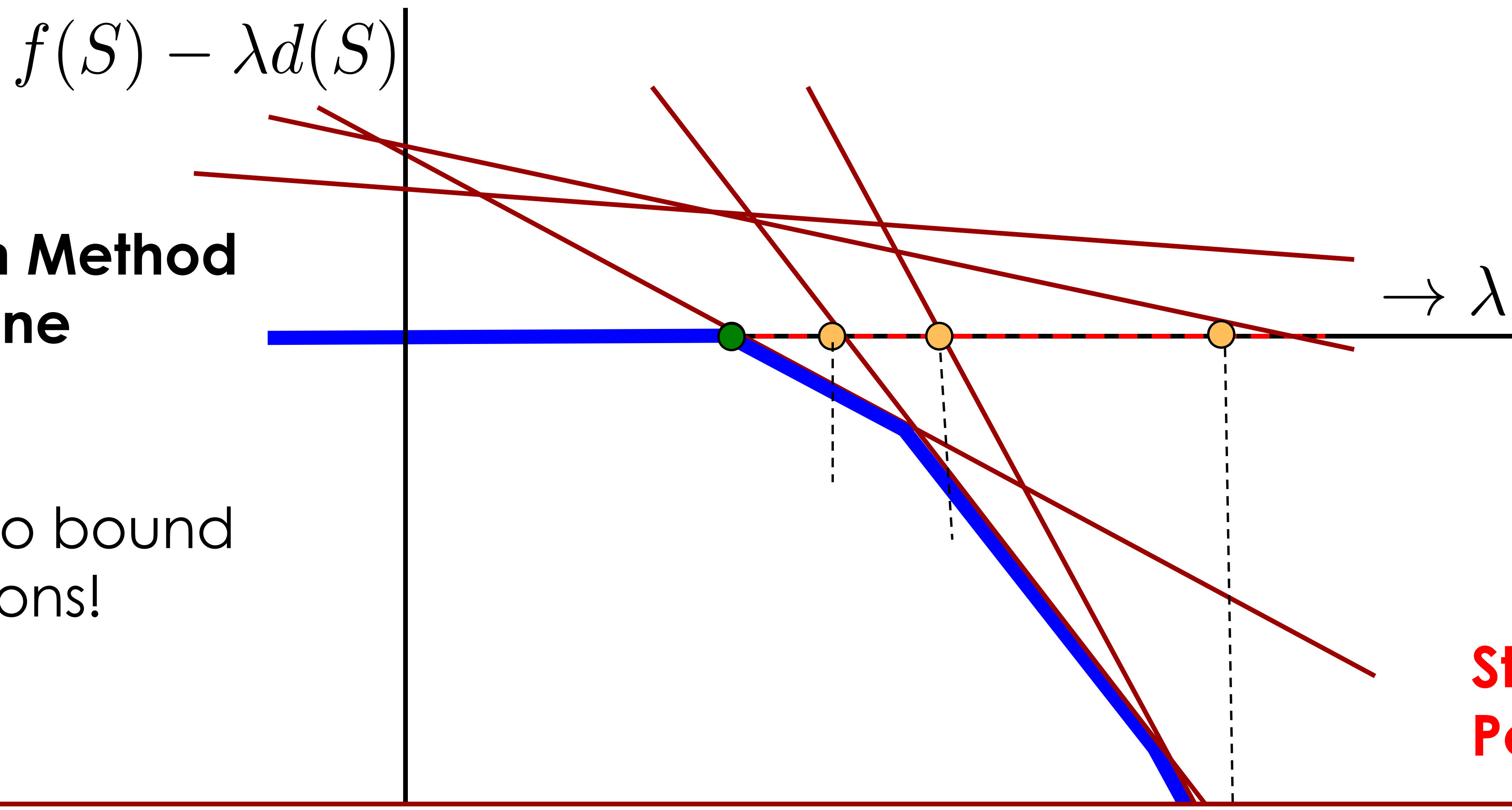
Recall

Inc-Fix uses only positive directions (well-understood)

General: Megiddo's parametric search: **Õ(n^8) SFM** [Nagano 2011]

$$\max \lambda : \lambda d \in P(f) \;\Rightarrow\; \max_{\lambda} : \min_{S \subseteq V} f(S) - \lambda d(S) \geq 0$$

Discrete Newton Method for Parametric Line Search



Open question to bound the no. of iterations!

Strongly Polynomial

We show a quadratic bound on the number of Newton's iterations:
<= **n^2 + o(n $\log^2$n) SFM** (n^6 improvement) [Goemans, Gupta, Jaillet, IPCO 2017]

Sequence of subsets

Consider any submodular function $f(\cdot)$
a sequence of sets $S_1, S_2, \cdots, S_q$

$$f_{\min} = -1, f(S_1) \geq 2,$$
$$f(S_k) \geq 4f(S_{k-1}) \text{ for } k > 1.$$

We show a quadratic bound on the number of Newton's iterations:
<= **n^2 + o(n $\log^2$n) SFM** (n^6 improvement) [Goemans, Gupta, Jaillet, IPCO 2017]

Sequence of subsets

Consider any submodular function $f(\cdot)$
a sequence of sets $S_1, S_2, \cdots, S_q$ ← How large can ***q*** be?

$$f_{\min} = -1, f(S_1) \geq 2,$$
$$f(S_k) \geq 4f(S_{k-1}) \text{ for } k > 1.$$

We show a quadratic bound on the number of Newton's iterations:
<= **n^2 + o(n $\log^2$n) SFM** (n^6 improvement) [Goemans, Gupta, Jaillet, IPCO 2017]

Sequence of subsets

Consider any submodular function $f(\cdot)$
a sequence of sets $S_1, S_2, \cdots, S_q$ ← How large can **q** be?

$$f_{\min} = -1, f(S_1) \geq 2,$$
$$f(S_k) \geq 4f(S_{k-1}) \text{ for } k > 1.$$

$S_k \notin \mathcal{R}(S_1, \cdots, S_{k-1})$ using submodularity of $f(\cdot)$

$$f(S \cap T) + f(S \cup T) \leq f(S) + f(T)$$

We show a quadratic bound on the number of Newton's iterations:
<= **n^2 + o(n $\log^2$n) SFM** (n^6 improvement) [Goemans, Gupta, Jaillet, IPCO 2017]

Sequence of subsets

Consider any submodular function $f(\cdot)$
a sequence of sets $S_1, S_2, \cdots, S_q$ ← How large can **q** be?

$$f_{\min} = -1, f(S_1) \geq 2,$$
$$f(S_k) \geq 4f(S_{k-1}) \text{ for } k > 1.$$

$S_k \notin \mathcal{R}(S_1, \cdots, S_{k-1})$ using submodularity of $f(\cdot)$

$q \leq \binom{n+1}{2} + 1$ using Birkhoff's representation theorem

We show a quadratic bound on the number of Newton's iterations:
<= **n^2 + o(n $\log^2$n) SFM** (n^6 improvement) [Goemans, Gupta, Jaillet, IPCO 2017]

Sequence of subsets

Consider any submodular function $f(\cdot)$

a sequence of sets $S_1, S_2, \cdots, S_q$ ← How large can **q** be?

$$f_{\min} = -1, f(S_1) \geq 2,$$

$$f(S_k) \geq 4f(S_{k-1}) \text{ for } k > 1.$$

$S_k \notin \mathcal{R}(S_1, \cdots, S_{k-1})$ using submodularity of $f(\cdot)$

$q \leq \binom{n+1}{2} + 1$ using Birkhoff's representation theorem

We show a quadratic bound on the number of Newton's iterations:
<= **n^2 + o(n $\log^2$n) SFM** (n^6 improvement) [Goemans, Gupta, Jaillet, IPCO 2017]

Outline

1. **Projections**
   - Motivation
   - Problem setup
   - **Novel algorithm**: Inc-Fix for separable convex minimization:
     - Main Result: O(n) SFM or O(n) Line searches
     - Exact computations, modulo solving a univariate equation

2. **Line Searches**
   - Previous best known: Megiddo's parametric search
   - Using Newton's Discrete Method: $n^2 + n \log^2 n$ SFM (n^6 improvement)

3. **What works best when**
   - Problems with Max-Cut and QUBO heuristics comparative studies
   - **Our framework**: Expanded instance library, Implementation of 37 heuristics, Large-scale cloud computing on the cross product
   - **Hyper-heuristic**: Map every instance to a feature space, learn "performance" of heuristics.

What works best when

Image from: https://www.dreamstime.com/stock-illustration-octopus-tools-illustration-image47507762

What works best when

Encounter
problem in practice

↓

Find out what is
known

↓

Run the “best” known
algorithm/heuristic
for the data

Image from: https://www.dreamstime.com/stock-illustration-octopus-tools-illustration-image47507762

What works best when

Encounter
problem in practice

↓

Find out what is
known

↓

Run the "best" known
algorithm/heuristic
for the data

From learning decisions,
to learning **performance of algorithms**

Image from: https://www.dreamstime.com/stock-illustration-octopus-tools-illustration-image47507762

Max-Cut: An NP-Hard Problem

Given an edge-weighted graph, partition nodes into two sets to maximize the weight of the edges between the sets

Equivalence with **Q**uadratic **U**nconstrained **B**inary **O**ptimization Problem (QUBO)

$$\max x^T Q x \quad x \in \{0,1\}^n$$

A lot of applications, and a lot of research!

- >32 published papers since 2010.

Computational experiments key to heuristic evaluation!

But hard to find which heuristic works best when

Max-Cut: An NP-Hard Problem

Given an edge-weighted graph, partition nodes into two sets to maximize the weight of the edges between the sets

Equivalence with **Q**uadratic **U**nconstrained **B**inary **O**ptimization Problem (QUBO)

$$\max x^T Q x \quad x \in \{0,1\}^n$$

A lot of applications, and a lot of research!

- >32 published papers since 2010.

Computational experiments key to heuristic evaluation!

But hard to find which heuristic works best when

Max-Cut: An NP-Hard Problem

Given an edge-weighted graph, partition nodes into two sets to maximize the weight of the edges between the sets

Equivalence with **Q**uadratic **U**nconstrained **B**inary **O**ptimization Problem (QUBO)

$$\max x^T Q x \quad x \in \{0,1\}^n$$

A lot of applications, and a lot of research!

- >32 published papers since 2010.

Computational experiments key to heuristic evaluation!

But hard to find which heuristic works best when

Max-Cut: An NP-Hard Problem

Given an edge-weighted graph, partition nodes into two sets to maximize the weight of the edges between the sets

Equivalence with **Q**uadratic **U**nconstrained **B**inary **O**ptimization Problem (QUBO)

$$\max x^T Q x \quad x \in \{0,1\}^n$$

A lot of applications, and a lot of research!

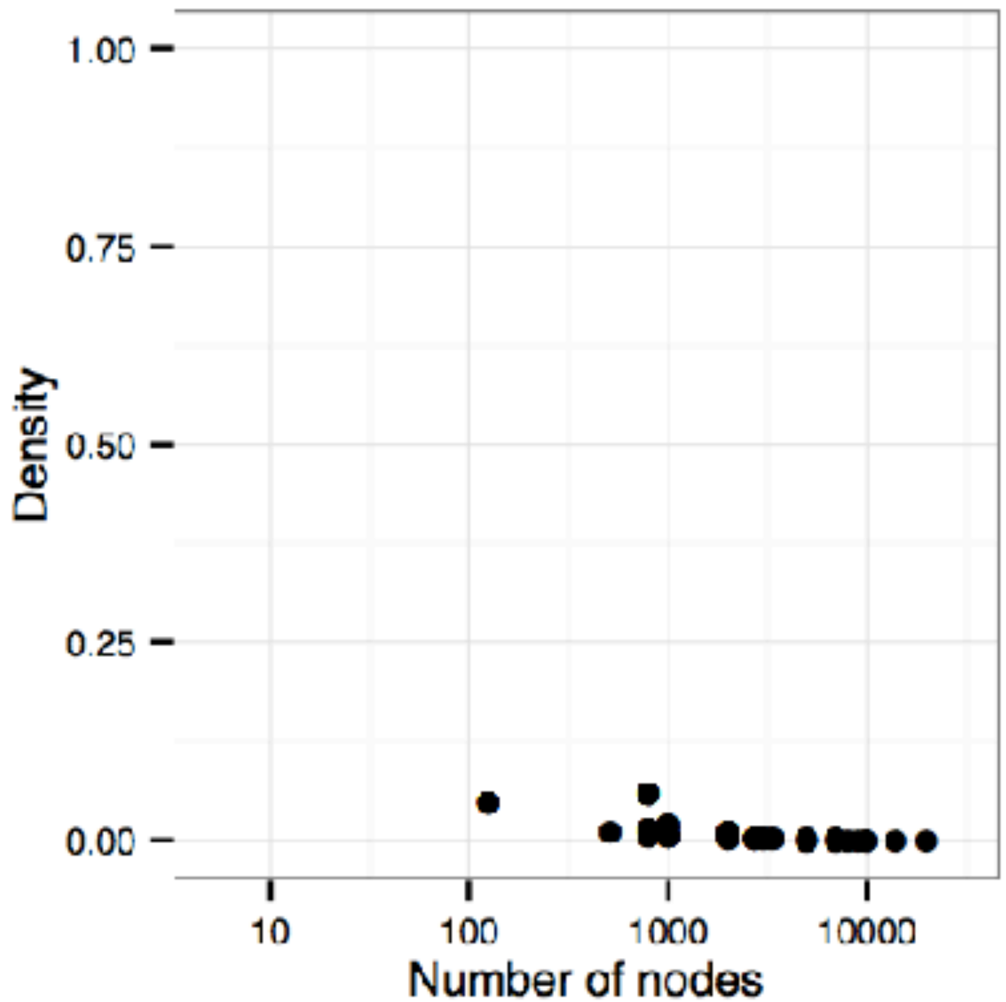
- ◇ >32 published papers since 2010.

Computational experiments key to heuristic evaluation!

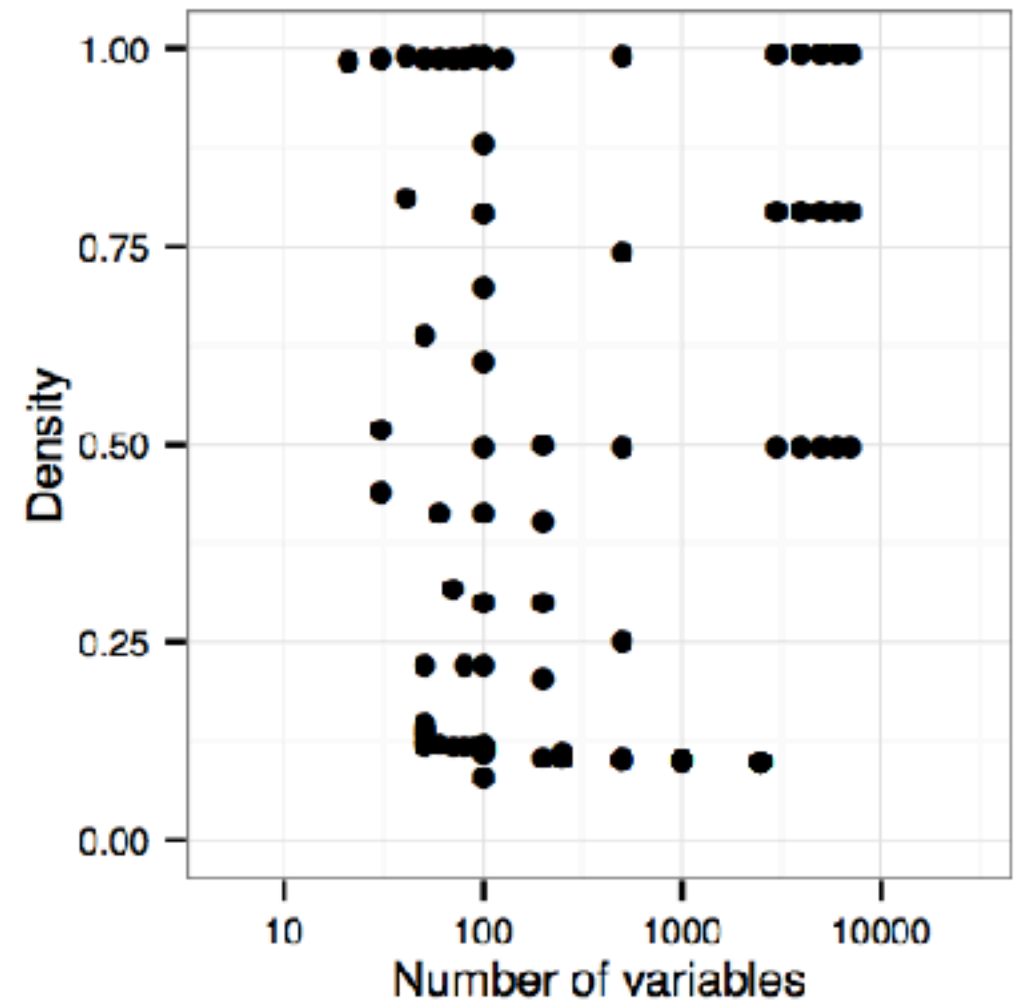
But hard to find which heuristic works best when

Problems with standard testbed

Homogeneous Test Bed: Max-Cut (105 graphs), QUBO (126 matrices)



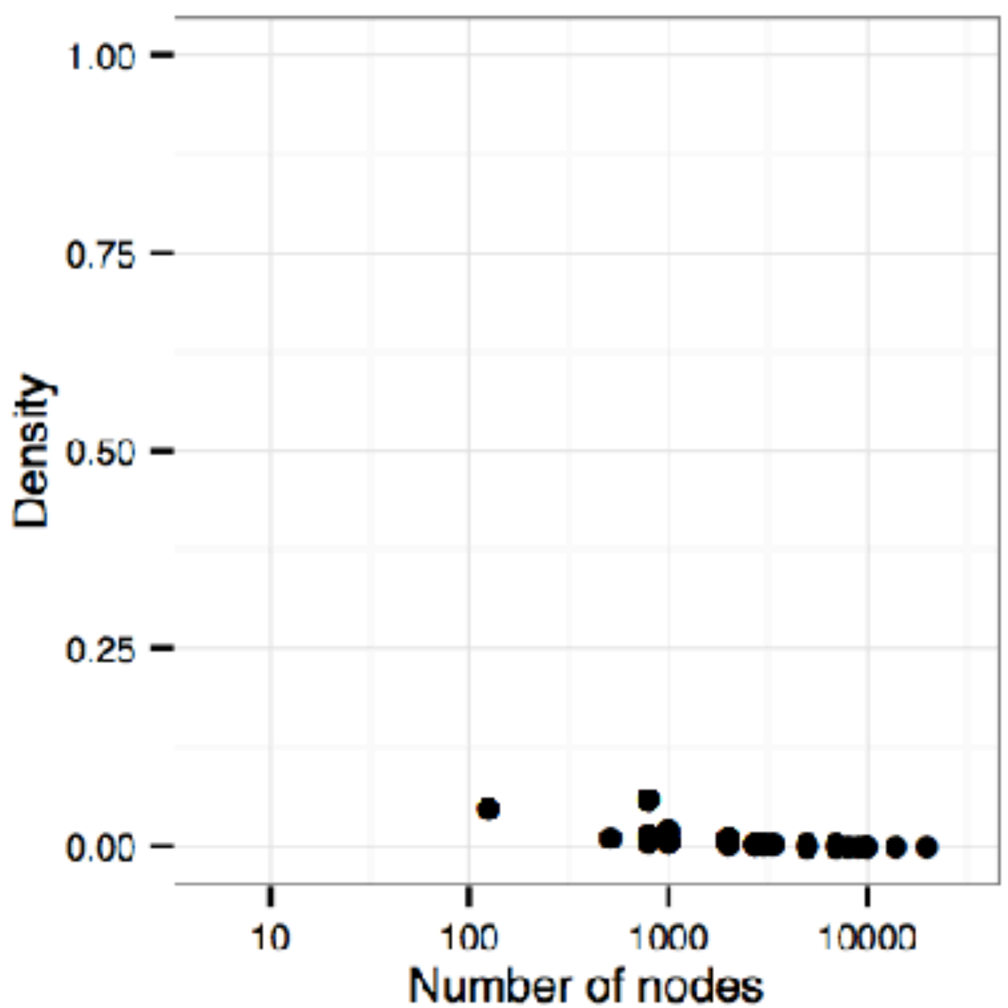
Max-Cut Instances



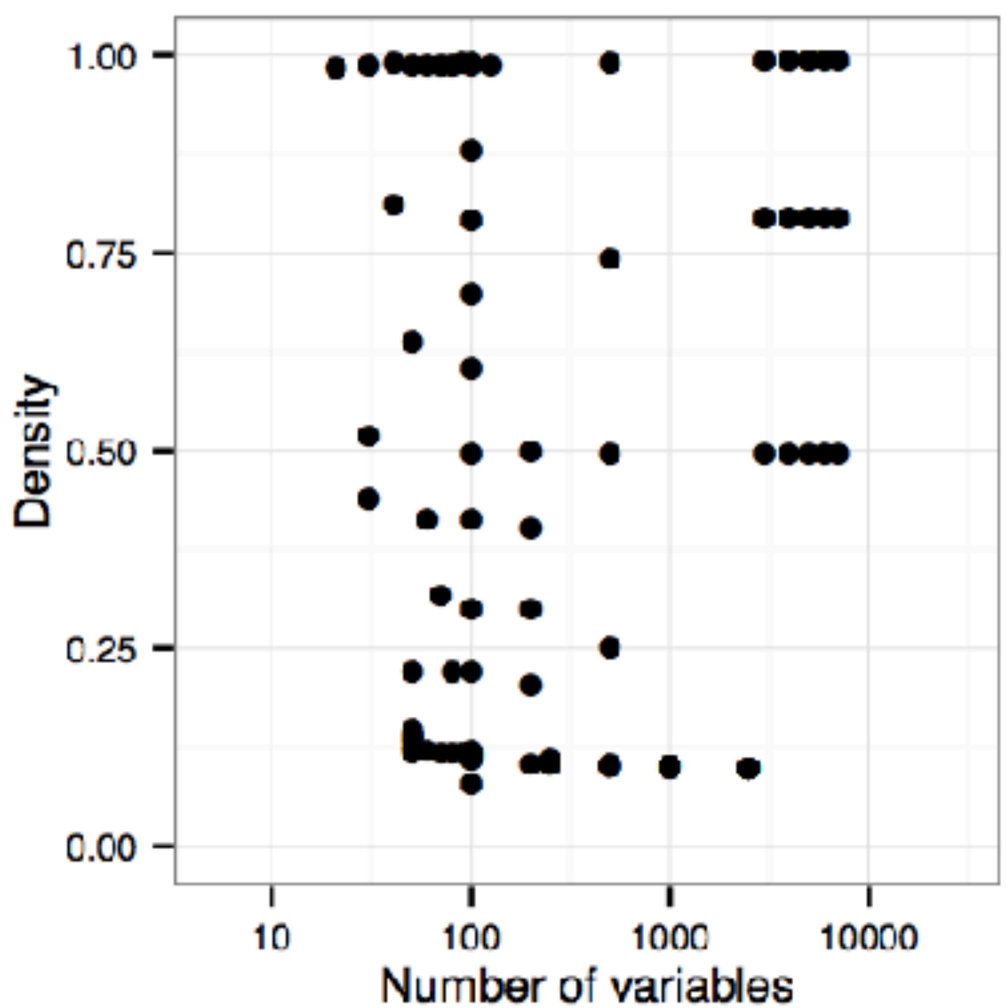
QUBO instances

Problems with standard testbed

Homogeneous Test Bed: Max-Cut (105 graphs), QUBO (126 matrices)



Max-Cut Instances

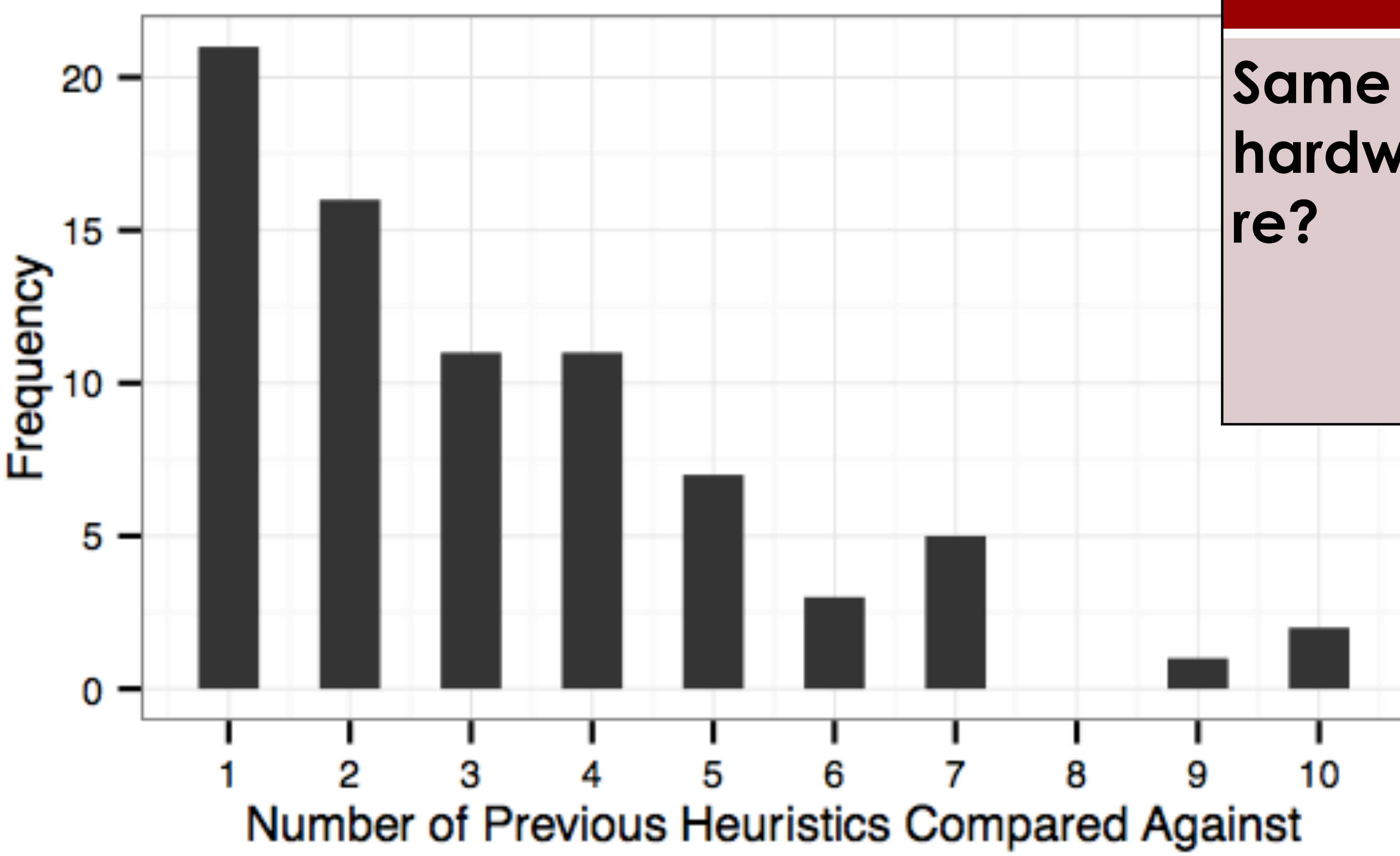


QUBO instances

Which Max-Cut heuristic works best for high density graphs?
Which QUBO heuristic works best for sparse matrices?

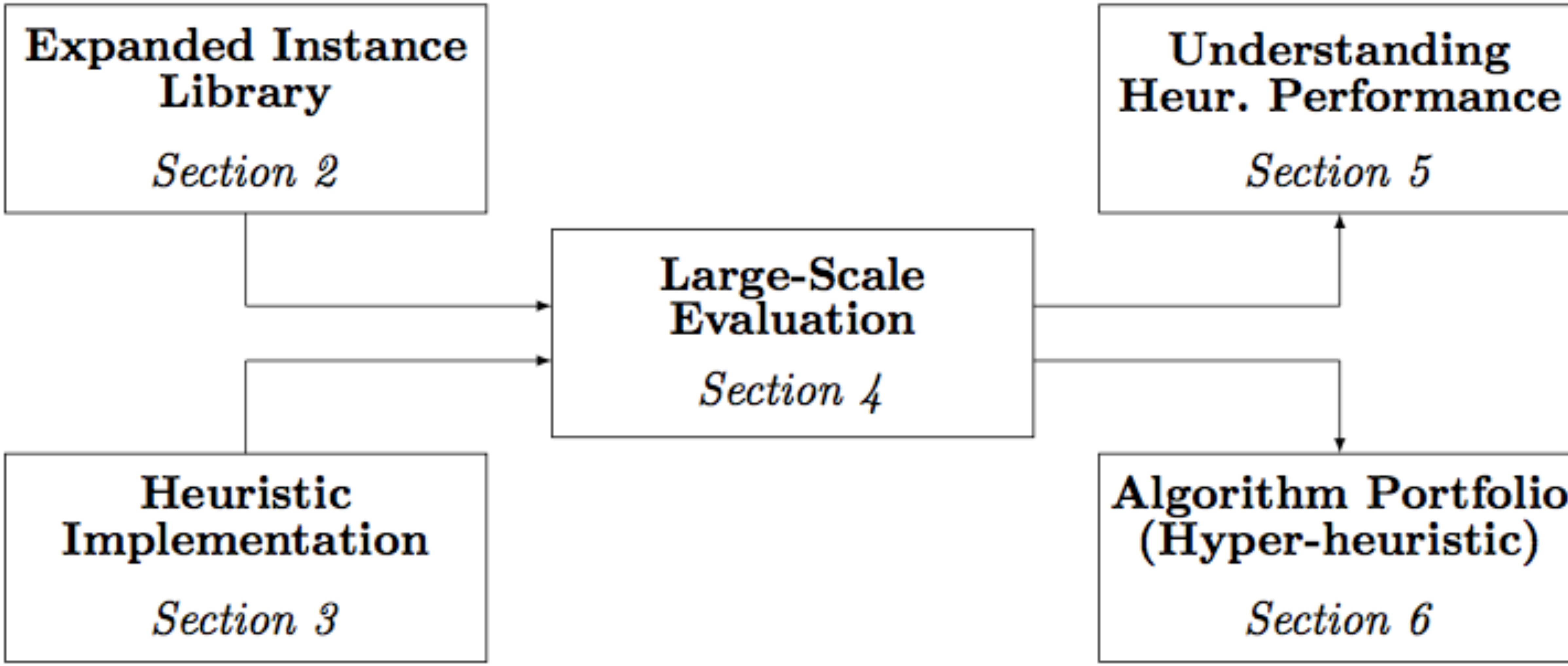
Problems with status-quo

- few published source code
- reimplementation uncommon
- different testing criteria
- comparison with small no. of heuristics…

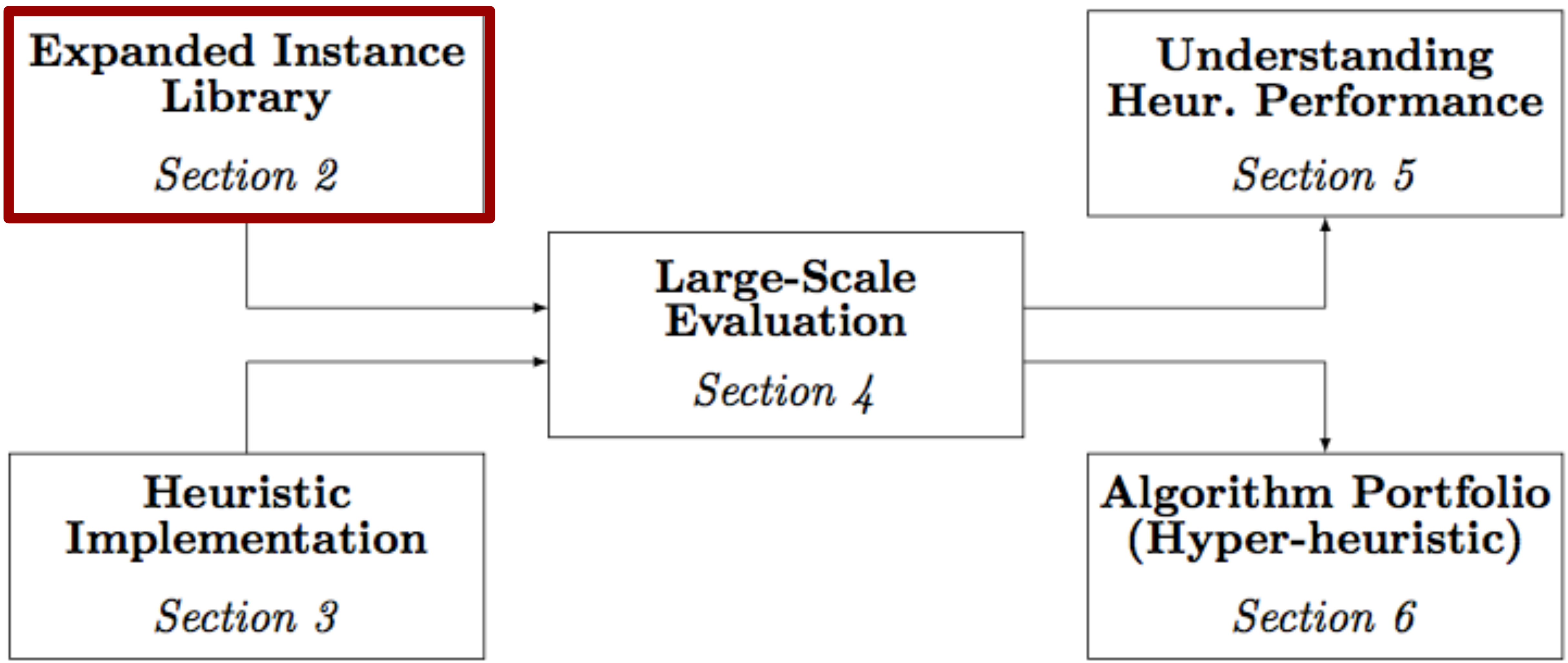


	Same runtime limit?		
Same hardware?		No	Yes
	No	55%	4%
	Yes	31%	**10%**

Our Approach

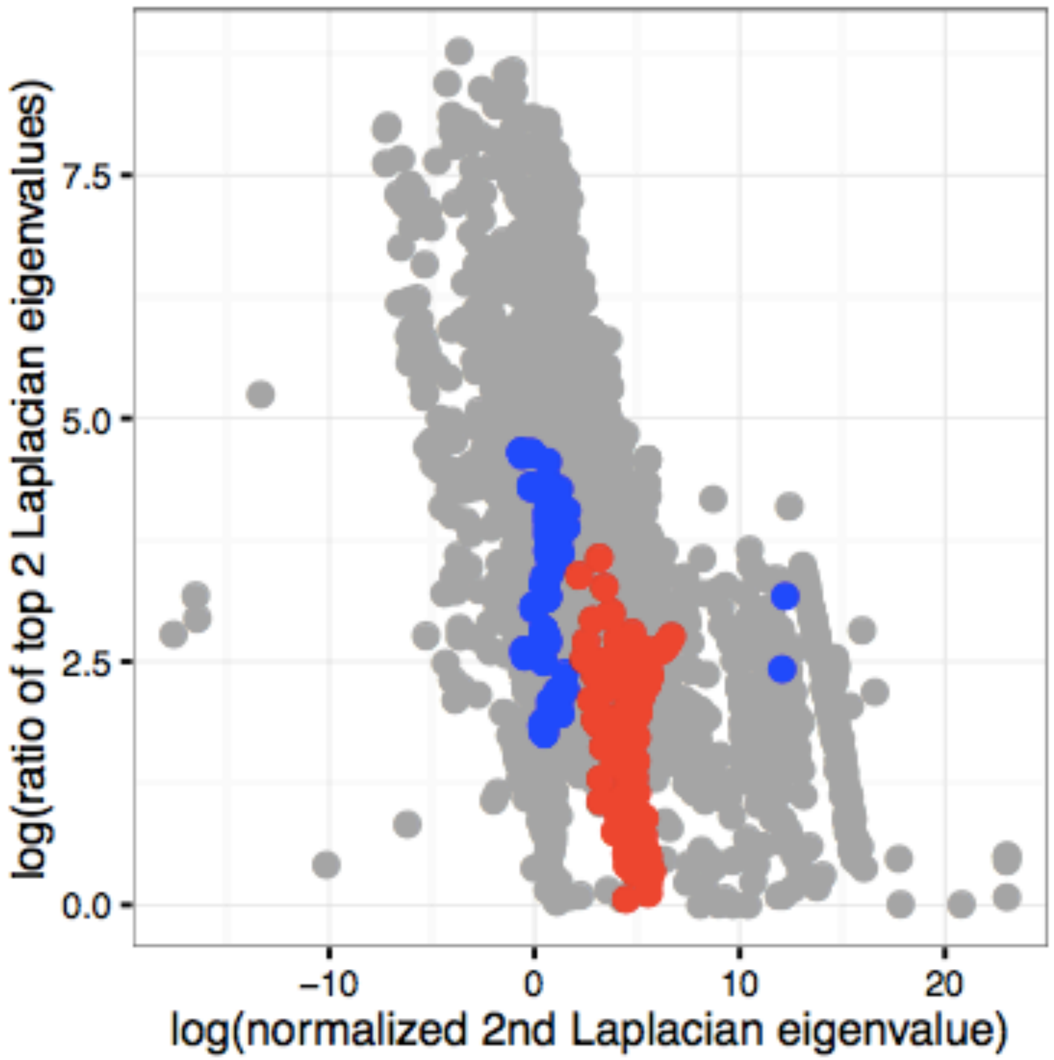


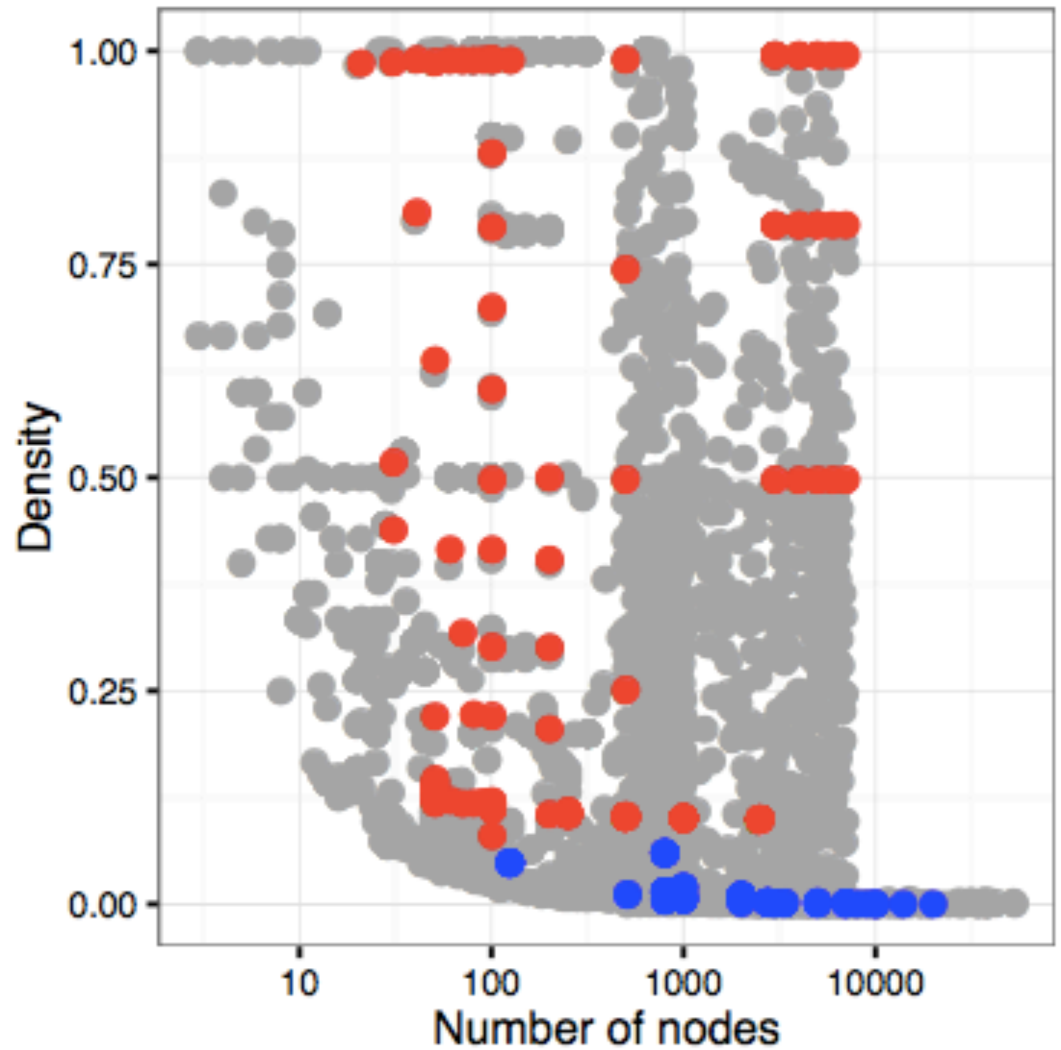
Our Approach



Expanded Instance Library

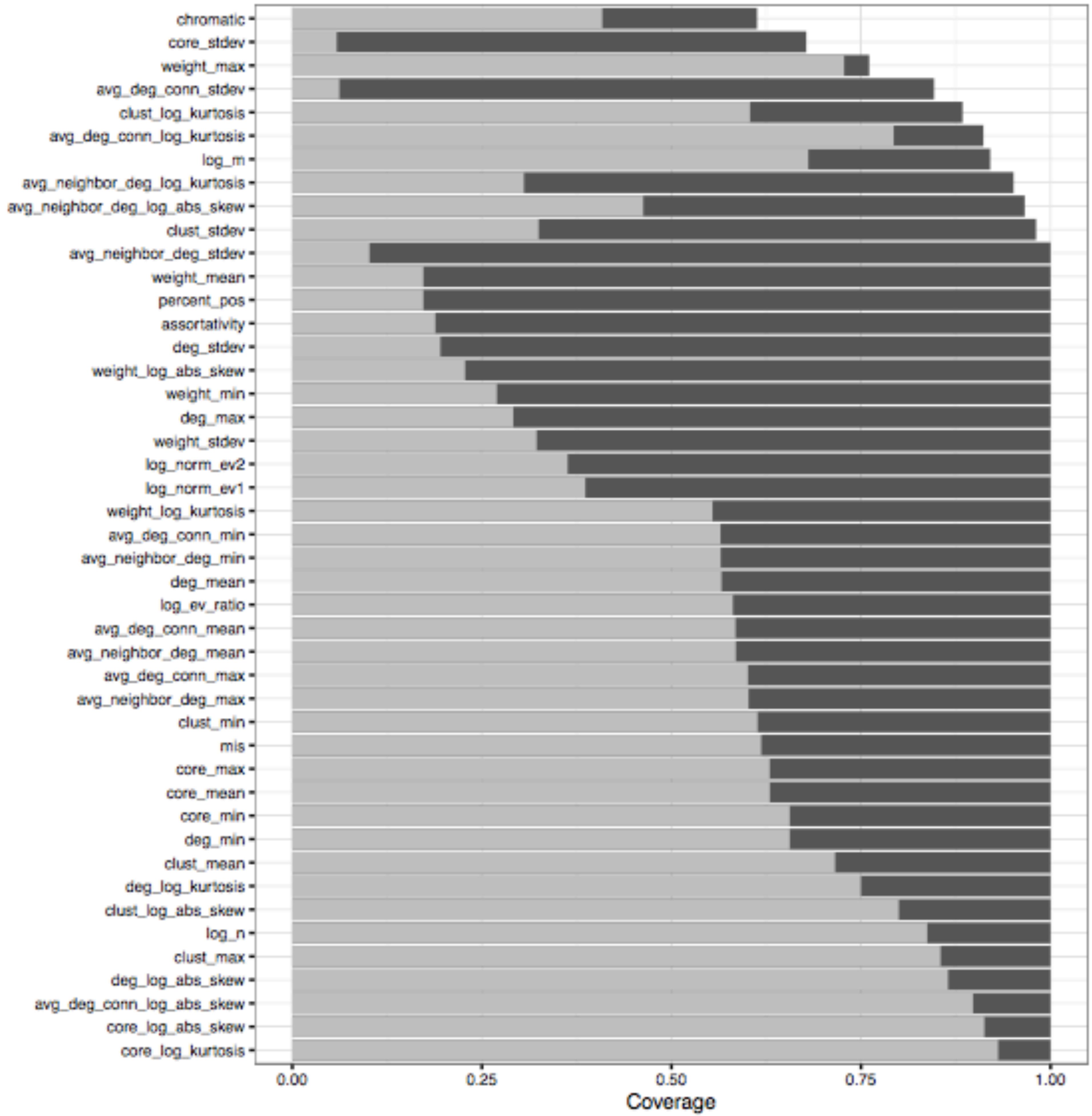
- Heterogeneous instances, capture instances in real instances
  - Real World Instances (tsplib, steinlib, dimacs, road networks, …)
  - Network science generators (ER, NWS, BA, …)
  - Sampled weights from 65 prob. distributions (uniform, weibull, …)





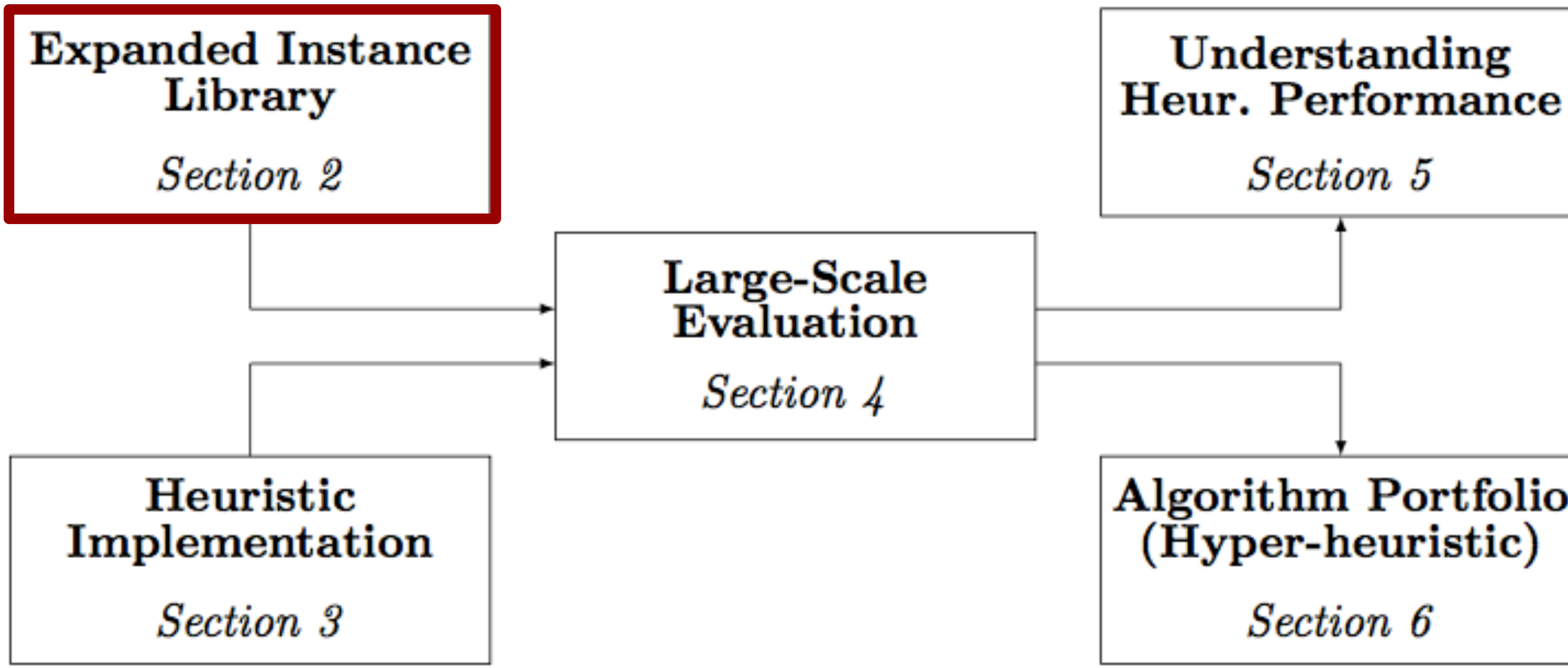
Heterogeneity: 58 Metrics

- 10 **global metrics**:
  - nodes, edges, 1st and 2nd eigenvalues of Laplacian, chromatic number, …
- 48 **local metrics** from summary statistics of edge/node attributes:
  - degree, avg. neighbor degree, clustering coefficient, core…
- Fast to compute – at most $O(n^2 \log n)$
- **Coverage** (for normalized metrics in [0,1]): union over all instances of a small interval around the metric value for each instance
  - average metric coverage for new test bed: 0.88 (interval +-0.05)
  - 0.31 for 95 std Max-Cut v/s 0.71 (0.69-0.77) for ~ 95 random new
  - 0.38 for 56 std QUBO v/s 0.64 (0.59-0.68) for ~56 random new

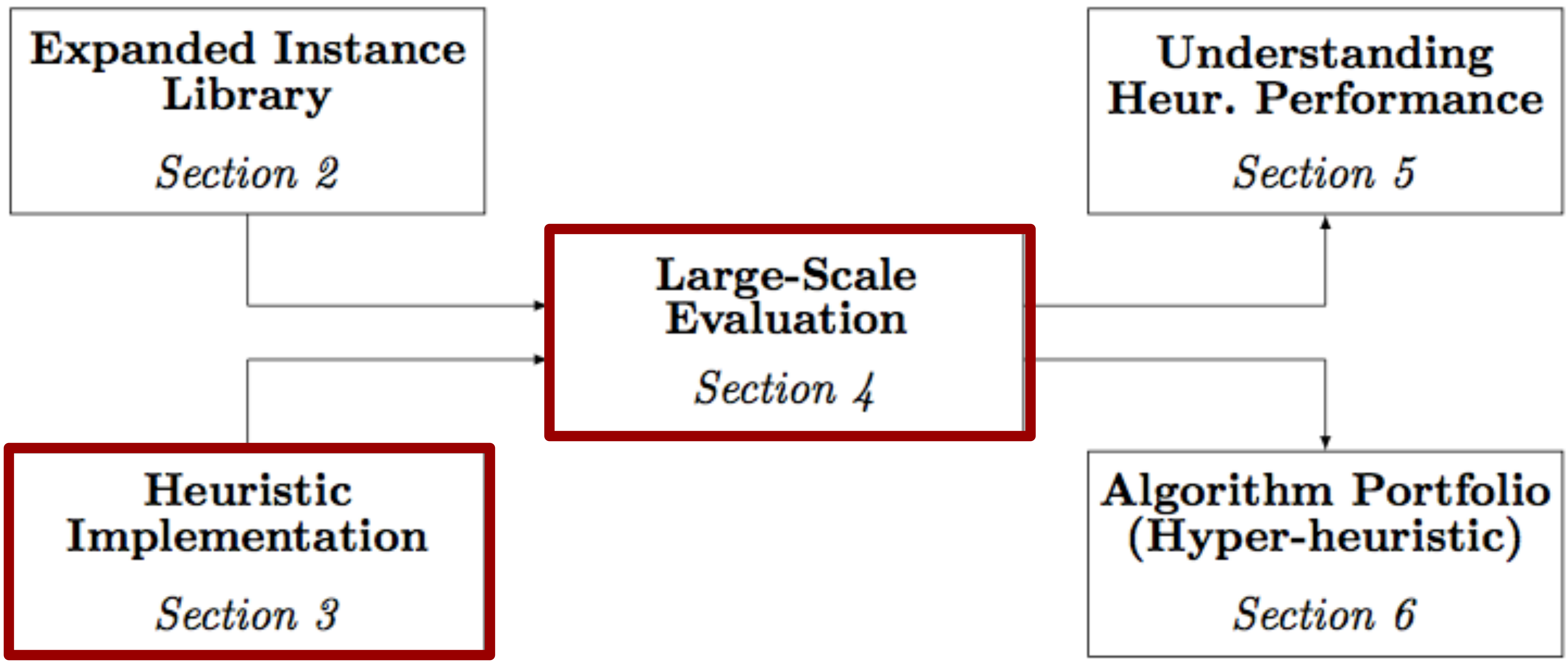


chromatic
core_stdev
weight_max
avg_deg_conn_stdev
clust_log_kurtosis
avg_deg_conn_log_kurtosis
log_m
avg_neighbor_deg_log_kurtosis
avg_neighbor_deg_log_abs_skew
clust_stdev
avg_neighbor_deg_stdev
weight_mean
percent_pos
assortativity
deg_stdev
weight_log_abs_skew
weight_min
deg_max
weight_stdev
log_norm_ev2
log_norm_ev1
weight_log_kurtosis
avg_deg_conn_min
avg_neighbor_deg_min
deg_mean
log_ev_ratio
avg_deg_conn_mean
avg_neighbor_deg_mean
avg_deg_conn_max
avg_neighbor_deg_max
clust_min
mis
core_max
core_mean
core_min
deg_min
clust_mean
deg_log_kurtosis
clust_log_abs_skew
log_n
clust_max
deg_log_abs_skew
avg_deg_conn_log_abs_skew
core_log_abs_skew
core_log_kurtosis
0.00
0.25
0.50
0.75
1.00
Coverage

Our Approach



Our Approach



Implementation + Evaluation

- **We did what one would expect**
  - thorough lit review (810 papers)
  - selected 95 papers (new heuristics)
  - implemented 37 heuristics from 19 highly cited papers

Implementation + Evaluation

- **We did what one would expect**
  - thorough lit review (810 papers)
  - selected 95 papers (new heuristics)
  - implemented 37 heuristics from 19 highly cited papers

- **Minor modifications to standardize:**
  - added random restarts
  - shared common code – data structures and subroutines
  - no parameter tuning

Implementation + Evaluation

- **We did what one would expect**
  - thorough lit review (810 papers)
  - selected 95 papers (new heuristics)
  - implemented 37 heuristics from 19 highly cited papers

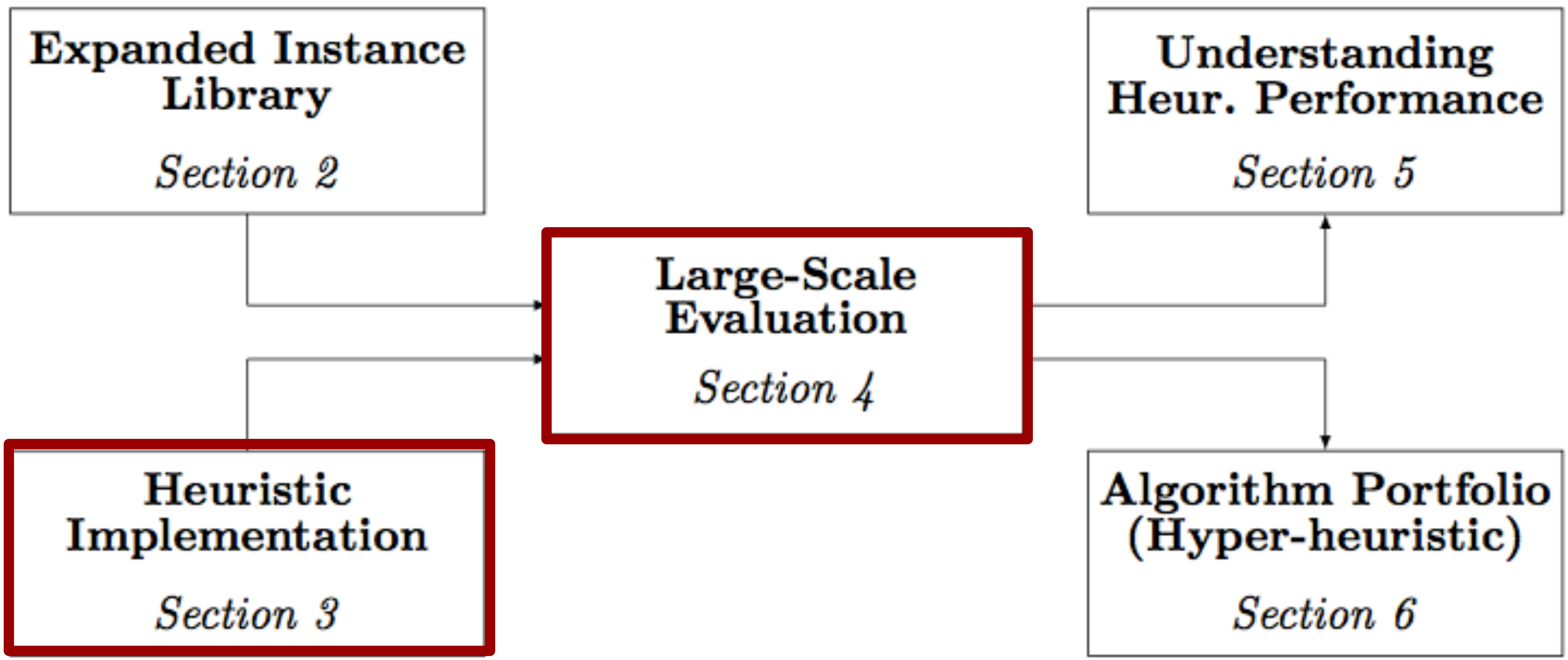
- **Minor modifications to standardize:**
  - added random restarts
  - shared common code – data structures and subroutines
  - no parameter tuning

- Cloud Computing – **Amazon EC2**
  - Instance specific runtime limit computation
    - too low: miss performance
    - too high: waste computational budget
  - any new heuristic can be tested for $32.5 (20.6 CPU days/ heuristic)

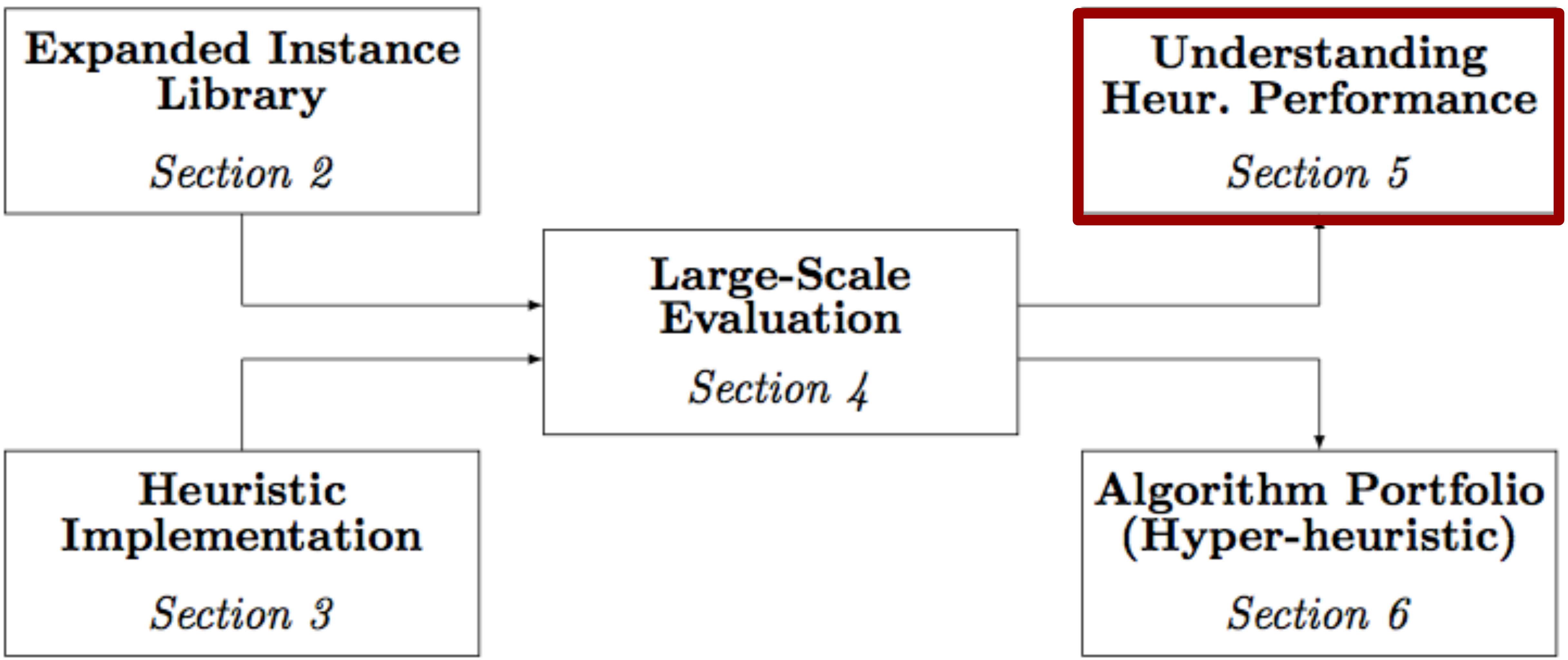
Open Source Code available at

https://github.com/MQLib/MQLib

Our Approach



Our Approach



Results

- **No heuristic dominated all the others**
  - 30/37 heuristics strictly best on at least one instance
  - No heuristic matched the best performance on more than 22.9% of the testbed
- **Standard test beds do not capture performance**
  - Example: GLS heuristic (Merz, Freisleben 1999)
  - Strictly best on no instances in the std test bed
  - Sole best-performing on 6.9% expanded test bed instances!

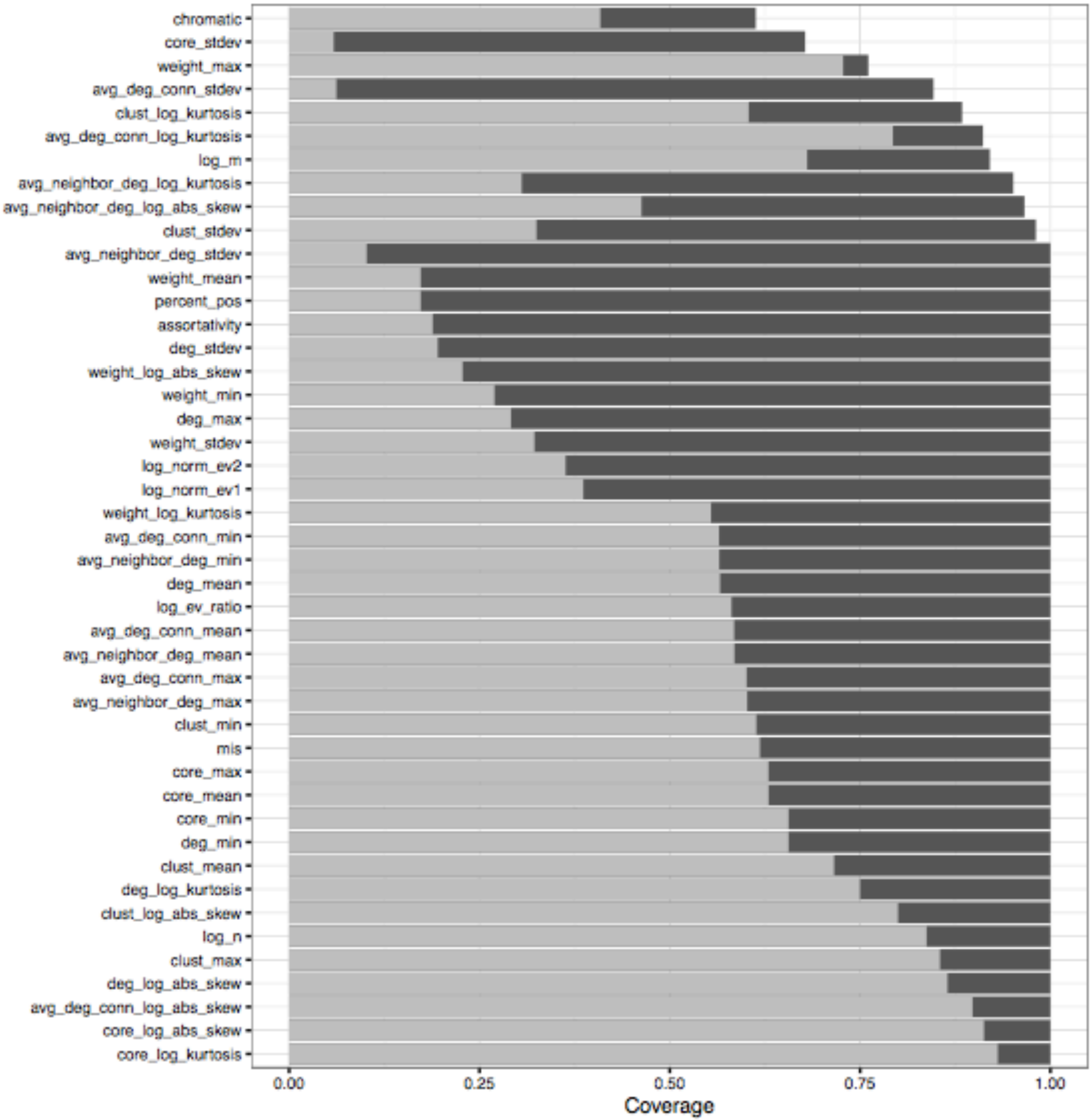
Heuristic	Instances with Top Performance (%) First-Equal (Mean-of-5)	First-Strict (Mean-of-5)	Best Achieved	Deviation from Best Solution (%) Worst-of-5 Deviation	Mean-of-5 Deviation	Best-of-5 Deviation	Avg. Rank
BUR02	**22.9**	**16.2**	32.7	**0.5**	**0.3**	**0.2**	10.7
FES02GVP	21.3	4.1	29.9	0.6	0.5	0.4	10.1
PAL04T3	19.4	2.3	30.5	0.8	0.6	0.5	**7.5**
FES02GP	19.3	5.5	27.1	0.9	0.7	0.6	14.2
FES02GV	18.0	1.5	26.0	0.8	0.7	0.6	11.2
PAL04T2	17.0	8.1	**33.3**	0.9	0.6	0.3	8.5
BEA98TS	16.4	5.0	22.4	2.6	2.4	2.1	16.6
LU10	14.9	2.7	23.9	1.7	1.5	1.2	13.1
FES02G	14.5	2.6	19.6	1.4	1.3	1.1	18.2
MER04	14.4	3.2	24.4	0.7	0.6	0.5	8.6
PAL04T1	14.0	2.7	21.7	2.4	2.2	1.9	15.2
MER99LS	12.8	6.9	30.9	0.7	0.5	0.3	10.0
MER02GRK	12.6	0.6	19.2	0.8	0.6	0.5	11.9
MER02LSK	11.3	0.4	19.3	1.0	0.8	0.7	13.3
PAL04T5	9.6	2.4	21.1	3.0	2.5	2.0	18.6
PAL06	9.4	0.9	21.9	2.4	1.9	1.4	17.3
ALK98	7.5	2.9	15.6	0.8	0.6	0.5	12.5
PAL04T4	7.3	0.2	18.4	3.8	3.2	2.6	21.7
GLO10	6.1	0.3	21.7	2.2	1.7	1.3	16.6
FES02V	6.0	0.3	15.1	1.3	1.1	0.9	13.8
KAT00	6.0	0.1	16.2	1.5	1.2	0.9	16.9
FES02VP	5.8	0.1	16.1	1.1	0.9	0.7	12.4
PAL04MT	5.1	0.2	16.8	4.0	3.4	2.8	24.1
MER02GR	3.5	0.0	5.7	3.2	3.0	2.8	24.3
GLO98	2.5	0.1	8.1	2.1	1.8	1.5	23.4
HAS00TS	2.2	0.4	7.2	1.6	1.3	1.1	20.3
HAS00GA	2.1	0.2	10.6	2.2	1.9	1.6	20.9
MER02LS1	1.8	0.0	6.3	3.0	2.8	2.6	25.1
PAR08	1.7	0.1	11.2	3.0	2.5	2.1	17.9
KAT01	1.6	0.6	4.0	2.7	2.4	2.1	26.3
DUA05	0.8	0.2	6.1	2.1	1.7	1.4	17.4
LOD99	0.7	0.0	3.4	5.7	5.3	4.9	30.2
LAG09HCE	0.4	0.3	4.1	2.3	1.9	1.6	20.4
BEA98SA	0.4	0.0	1.3	4.8	4.0	3.2	30.8
MER99CR	0.0	0.0	0.0	15.9	14.4	12.9	34.6
MER99MU	0.0	0.0	0.0	24.8	23.3	21.7	35.7
LAG09CE	0.0	0.0	0.0	30.8	30.1	29.2	36.9

Table 1 Summary of results for each heuristic across the 2,635 interesting instances. The best heuristic under

Can we predict which heuristic would work best on an unseen data instance?

"the algorithm selection problem is to learn the mapping from instance features to the best algorithm to run on an instance"

— Rice (1976)



… **Phase transitions**
(Cheeseman et al. 1991, Hartman and Weigt, 2006)

… **Landscape analysis**
(Stadler and Schnabl 1992,
Krzkakala et al. 2004,
Hartman and Weigt 2003,
Gent and Walsh 1996,
Smith-miles et al. 2010,
Wang et al. 2013…)

Interpreting Heuristic Performance

[… "algorithmic footprints" Smith-Miles et al. 2014]

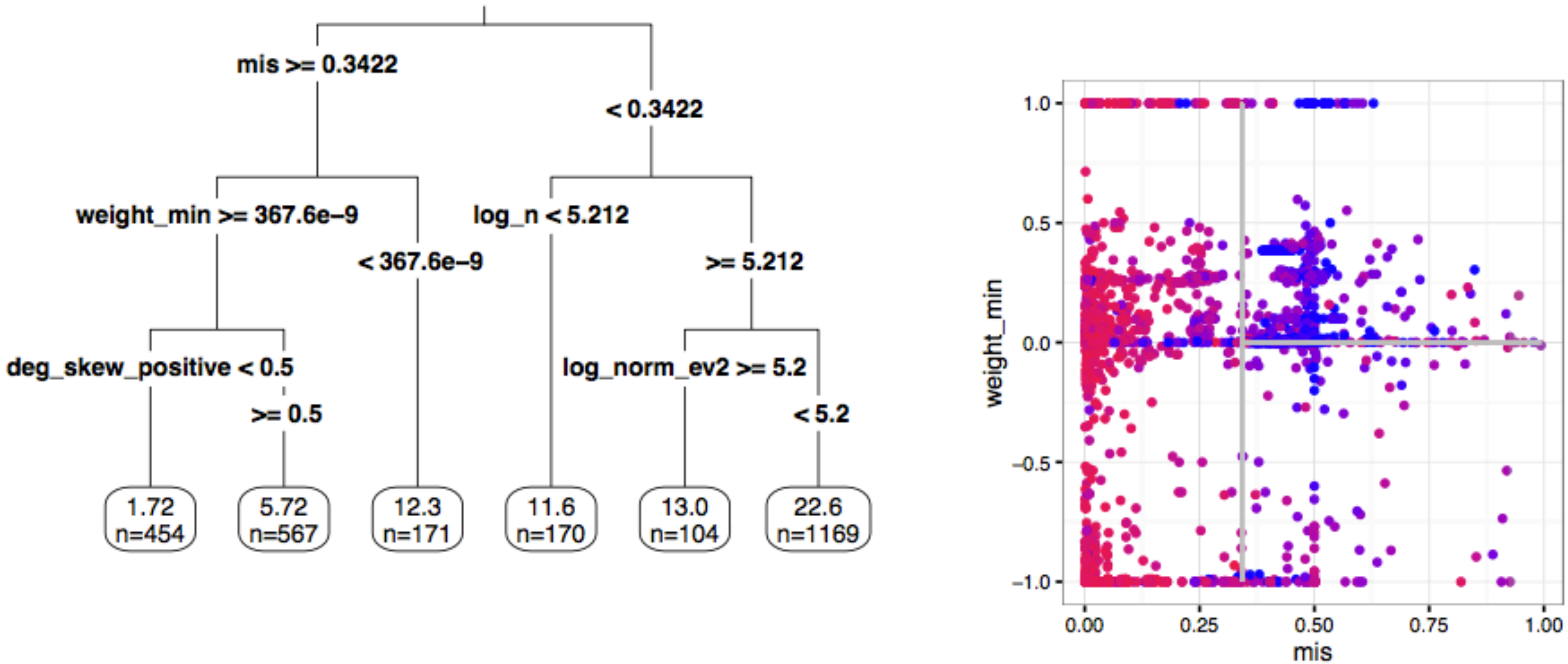


Figure 9: A CART model identifying instances on which FES02GP performs particularly well or poorly. Blue indicates the heuristic performed well (rank near 1) and red indicates the heuristic performed poorly (rank near 37).

Interpreting Heuristic Performance

[… "algorithmic footprints" Smith-Miles et al. 2014]

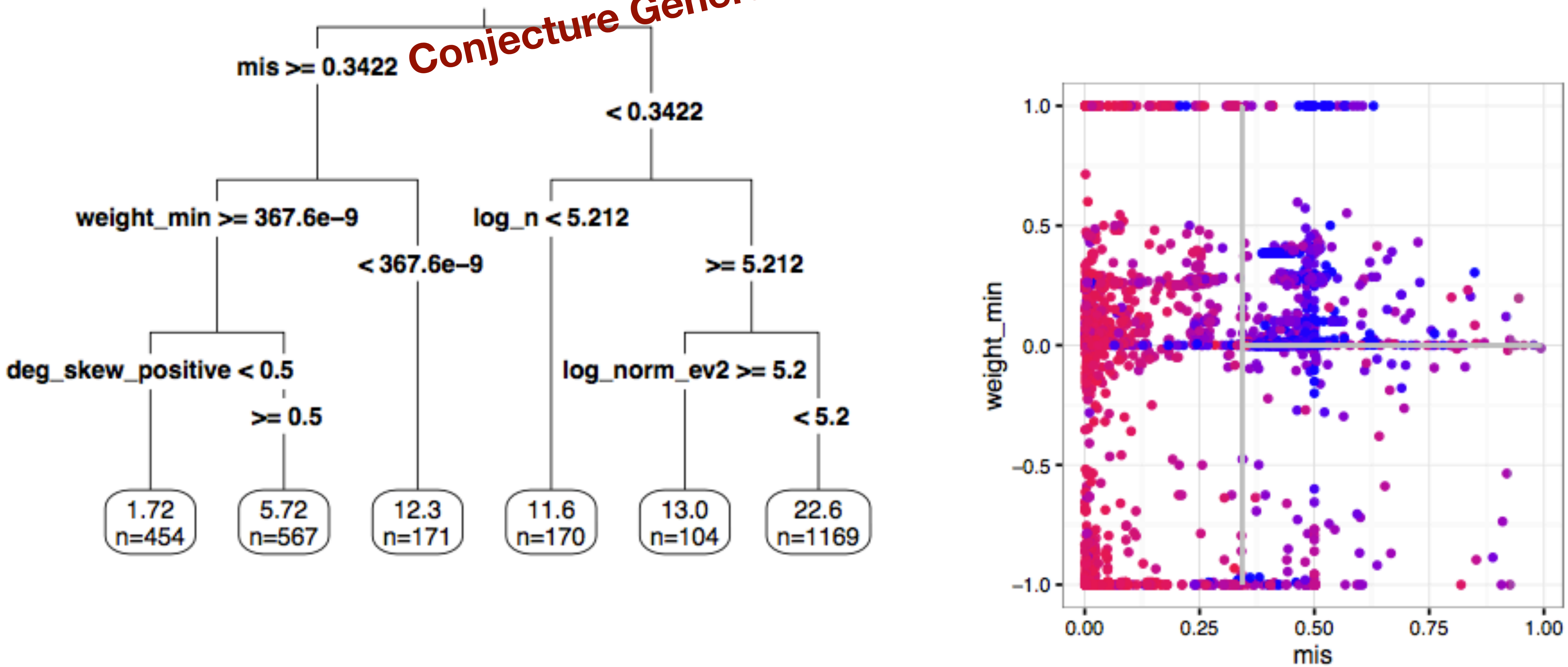
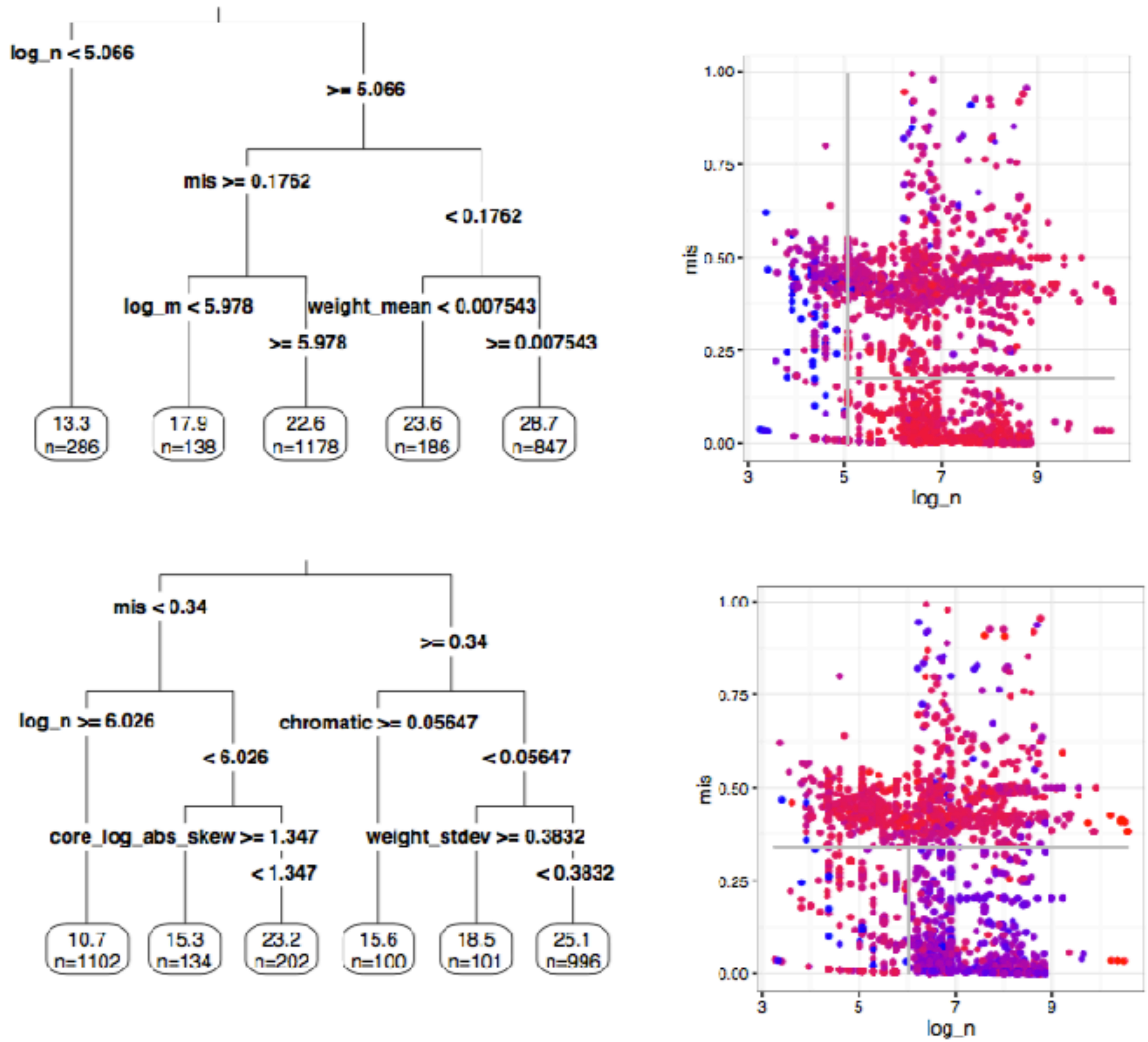


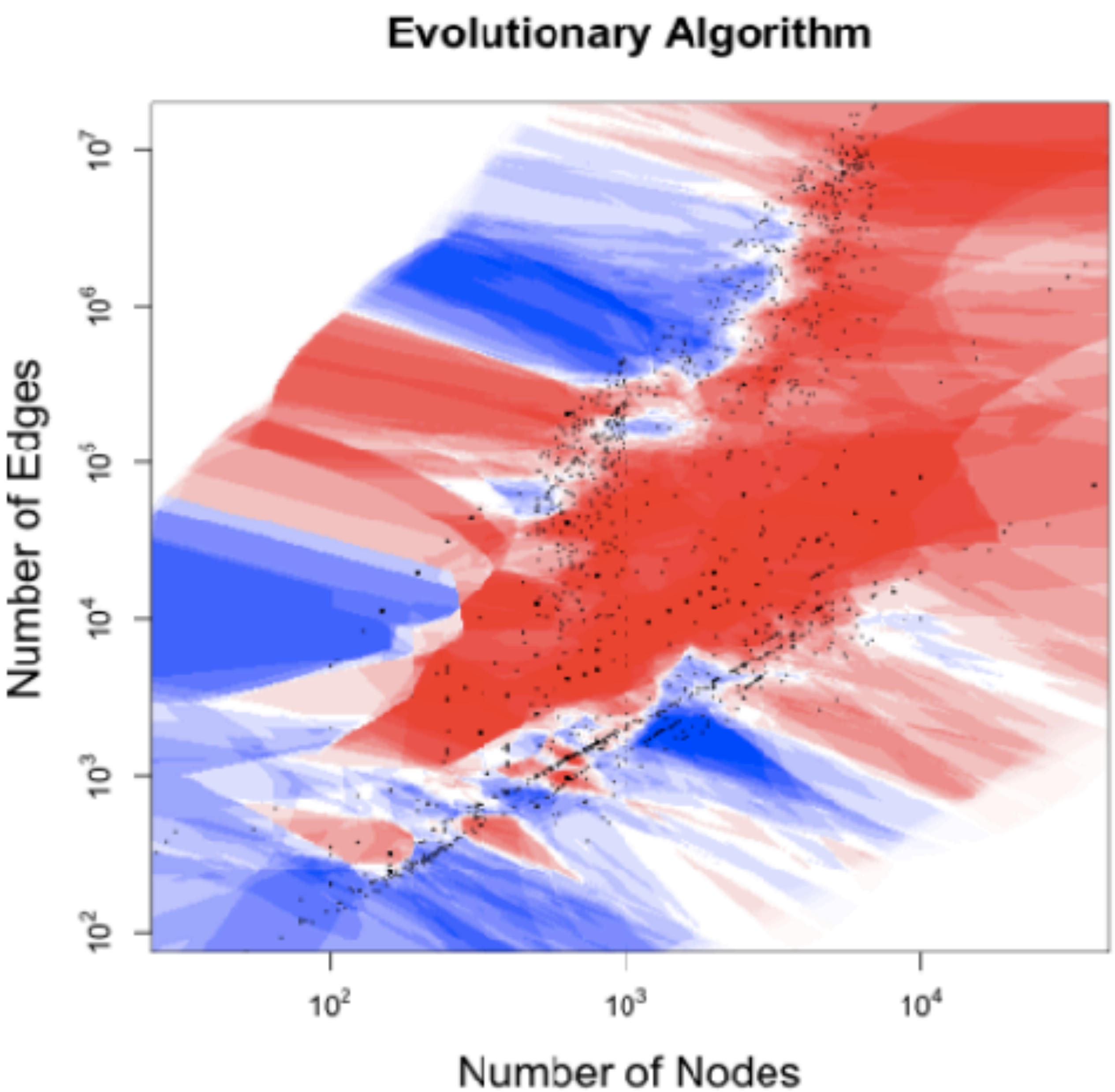
Figure 9: A CART model identifying instances on which FES02GP performs particularly well or poorly. Blue indicates the heuristic performed well (rank near 1) and red indicates the heuristic performed poorly (rank near 37).

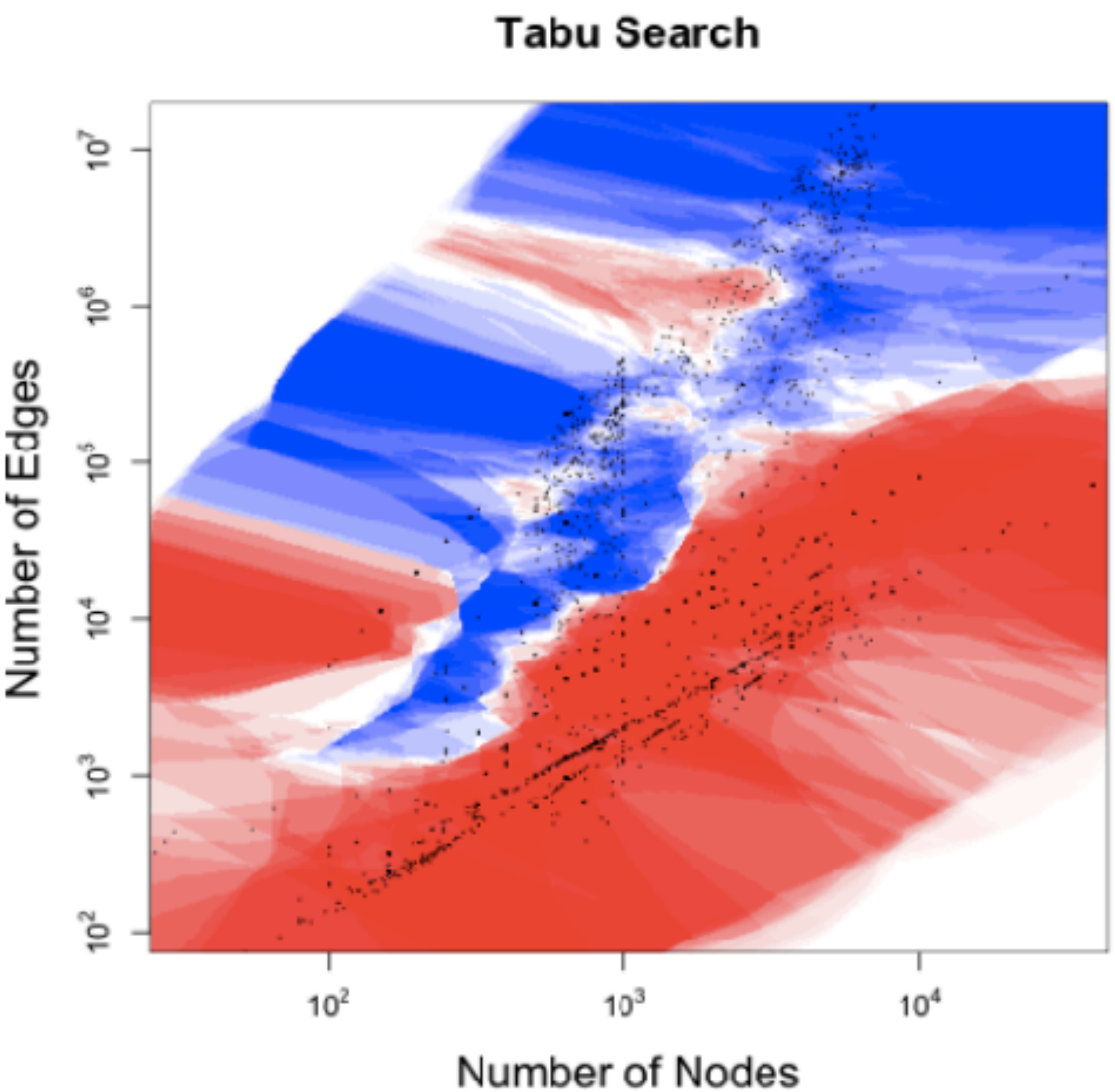
Comparing Heuristic Performance



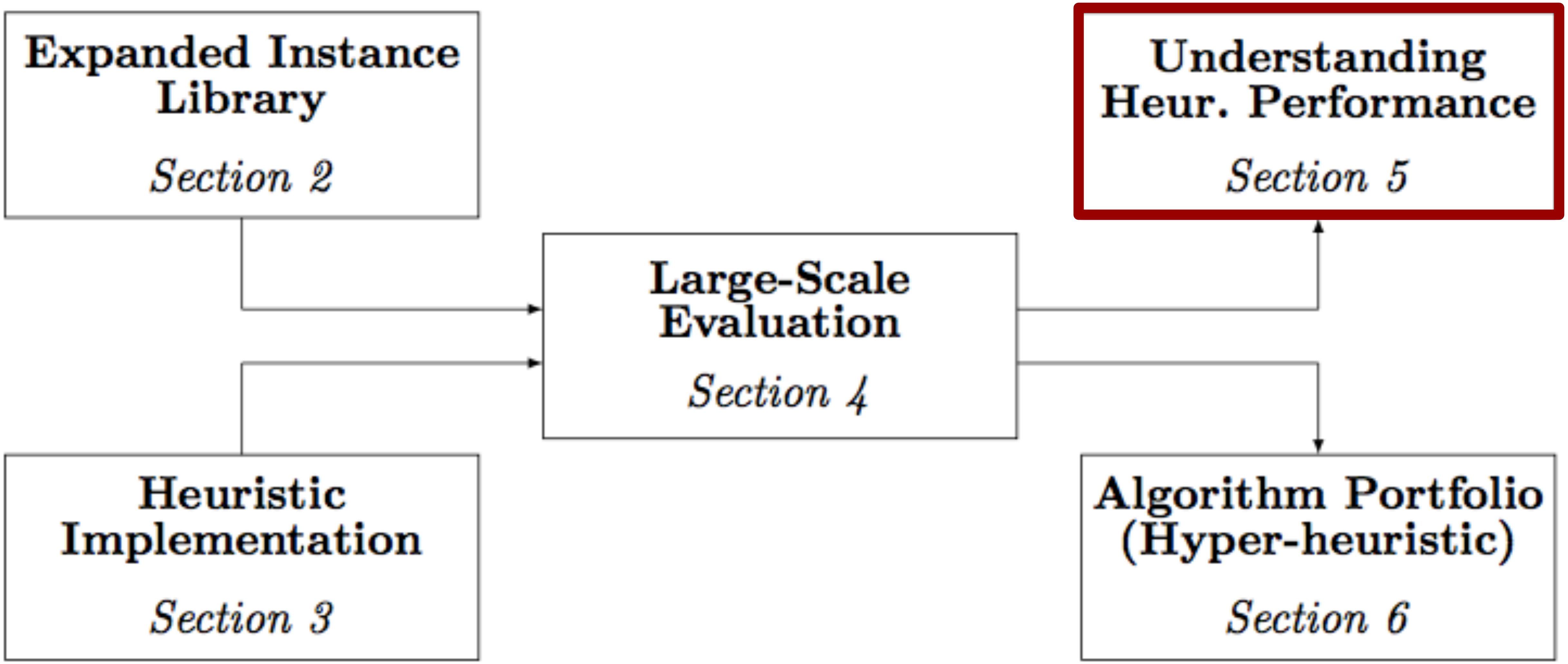
Interpretable models identifying the instances on which GLO98 (top) and PAR08 (bottom) perform

Heuristic Class Performance

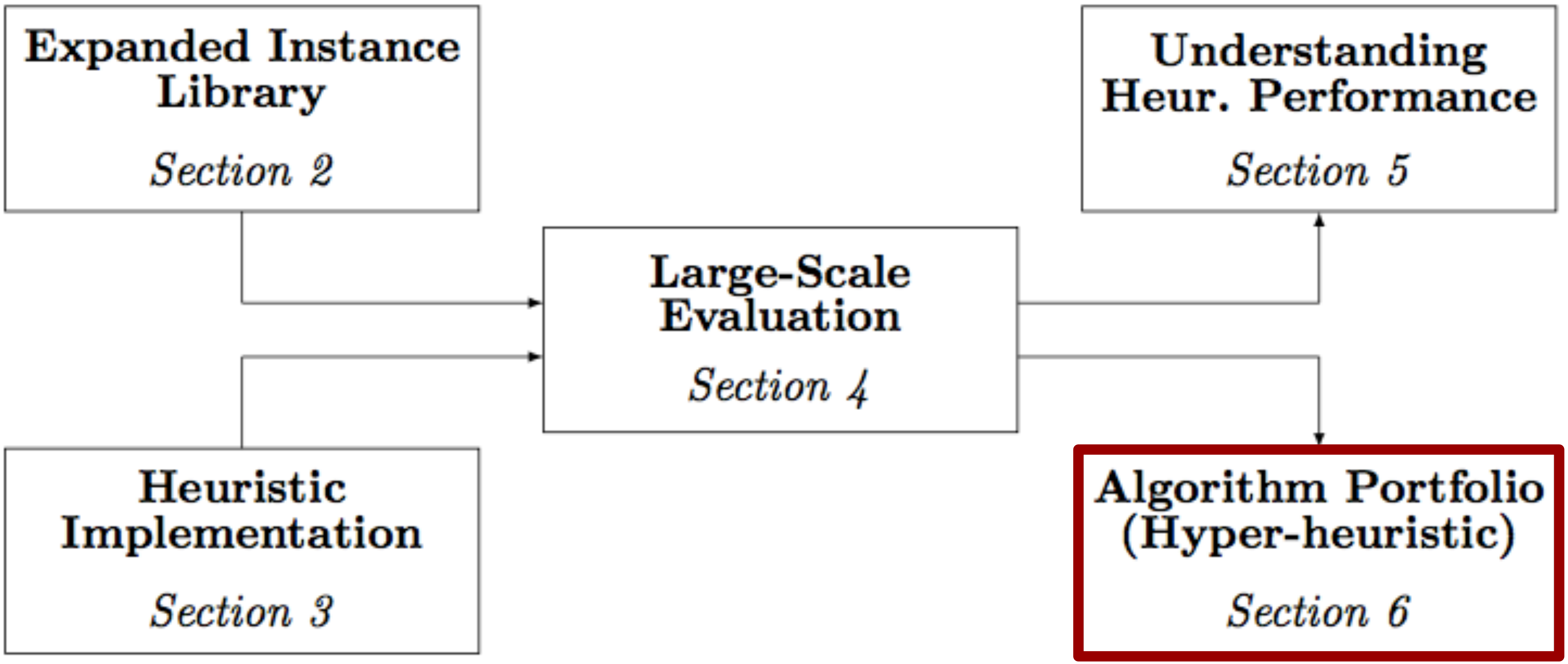




Our Approach



Our Approach



Algorithm Portfolio or Hyper-heuristic

[… SAT solvers (Xu et al 2008), constrained prog (O' Mahoney et al. 2008)]

- **Random Forest Model** for each heuristic
  - Predicts if it will obtain the best solution using 58 features
  - Final heuristic selected has maximum predicted probability
  - Small fraction of runtime budget to select heuristic and then run the selected heuristic on remaining time

- **Represents state-of-the-art Max-Cut and QUBO heuristic!**
  - Improves significantly over best single heuristic (BUR02):
    - Probability of obtaining best solution: increased from **15% to 37%**
    - Avg. deviation from best solution reduced from **0.34% to 0.09%**
    - Running 8 heuristics in parallel: **48%** best solution, **0.05%** avg. dev.

(joint work with Iain Dunning, John Silberholz. INFORMS Journal on Computing, 2017)

Outline

1. **Projections**
   - Motivation
   - Problem setup
   - **Novel algorithm**: Inc-Fix for separable convex minimization:
     - Main Result: O(n) SFM or O(n) Line searches
     - Exact computations, modulo solving a univariate equation

swatig@alum.mit.edu
swatigupta.tech

2. **Line Searches**
   - Previous best known: Megiddo's parametric search
   - Using Newton's Discrete Method: $n^2 + n \log^2 n$ SFM (n^6 improvement)

3. **What works best when**
   - Problems with Max-Cut and QUBO heuristics comparative studies
   - **Our framework**: Expanded instance library, Implementation of 37 heuristics, Large-scale cloud computing on the cross product
   - **Hyper-heuristic**: Map every instance to a feature space, learn "performance" of heuristics

Paper	Type	Short name	Description
Alkhamis et al. (1998)	Q	ALK98	Simulated annealing
Beasley (1998)	Q	BEA98SA	Simulated annealing
		BEA98TS	Tabu search
Burer et al. (2002)	M	BUR02	Non-linear optimization with local search
Duarte et al. (2005)	M	DUA05	Genetic algorithm with VNS as local search
Festa et al. (2002)	M	FES02G	GRASP with local search
		FES02GP	GRASP with path-relinking
		FES02V	VNS
		FES02VP	VNS with path-relinking
		FES02GV	GRASP with VNS local search
		FES02GVP	GRASP & VNS with path-relinking
Glover et al. (1998)	Q	GLO98	Tabu search
Glover et al. (2010)	Q	GLO10	Tabu search with long-term memory
Hasan et al. (2000)	Q	HAS00GA	Genetic algorithm
		HAS00TS	Tabu search
Katayama et al. (2000)	Q	KAT00	Genetic algorithm with k-opt local search
Katayama and Narihisa (2001)	Q	KAT01	Simulated annealing
Laguna et al. (2009)	M	LAG09CE	Cross-entropy method
		LAG09HCE	Cross-entropy method with local search
Lodi et al. (1999)	Q	LOD99	Genetic algorithm
Lü et al. (2010)	Q	LU10	Genetic algorithm with tabu search
Merz and Freisleben (1999)	Q	MER99LS	Genetic algorithm, with crossover and local search
		MER99MU	Genetic algorithm, with mutation only
		MER99CR	Genetic algorithm, with crossover only
Merz and Freisleben (2002)	Q	MER02GR	GRASP without local search
		MER02LS1	1-opt local search with random restarts
		MER02LSK	k-opt local search with random restarts
		MER02GRK	k-opt local search with GRASP
Merz and Katayama (2004)	Q	MER04	Genetic algorithm, with k-opt local search
Palubeckis (2004)	Q	PAL04T1	Tabu search
		PAL04T2	Iterated tabu search
		PAL04T3	Tabu search with GRASP
		PAL04T4	Tabu search with long-term memory
		PAL04T5	Iterated tabu search
		PAL04MT	Tabu search
Palubeckis (2006)	Q	PAL06	Iterated tabu search
Pardalos et al. (2008)	Q	PAR08	Global equilibrium search

variable	MDGr	MDG	pct	variable	MDGr	MDG	pct
log_n	7.90	23.00	74.00	log_m	8.10	13.00	74.00
log_norm_ev2	8.50	12.40	68.00	log_ev_ratio	10.00	10.10	71.00
log_norm_ev1	10.90	9.00	65.00	weight_log_kurtosis	11.40	11.30	53.00
weight_min	13.00	10.80	53.00	weight_mean	13.40	9.90	47.00
weight_stdev	13.40	8.40	47.00	deg_stdev	14.40	7.60	47.00
weight_log_abs_skew	16.60	6.80	32.00	mis	18.60	8.90	29.00
assortativity	18.70	7.60	26.00	avg_deg_conn_min	19.90	6.30	21.00
avg_deg_conn_stdev	20.20	5.30	26.00	deg_log_abs_skew	20.40	5.60	21.00
deg_max	21.50	6.80	15.00	avg_deg_conn_log_kurtosis	21.80	6.20	26.00
avg_neighbor_deg_min	22.30	5.80	15.00	chromatic	23.10	5.50	6.00
deg_log_kurtosis	23.60	5.10	12.00	avg_neighbor_deg_stdev	24.70	5.30	24.00
avg_deg_conn_max	25.40	3.50	12.00	avg_deg_conn_mean	25.40	3.80	6.00
avg_neighbor_deg_mean	25.40	3.90	12.00	core_mean	25.50	4.80	18.00
avg_neighbor_deg_max	26.10	3.80	15.00	deg_mean	26.40	4.20	12.00
core_stdev	26.60	4.50	6.00	deg_min	26.80	3.60	9.00
avg_neighbor_deg_log_abs_skew	27.30	4.20	0.00	avg_neighbor_deg_log_kurtosis	27.40	4.10	6.00
core_log_kurtosis	27.90	4.40	9.00	clust_stdev	28.00	4.60	6.00
core_min	28.40	3.50	6.00	clust_mean	28.60	5.10	3.00
avg_deg_conn_log_abs_skew	29.10	3.90	9.00	core_log_abs_skew	30.50	3.90	3.00
clust_max	31.10	6.00	6.00	core_max	33.10	2.90	0.00
clust_log_abs_skew	33.40	3.60	0.00	percent_pos	34.80	3.60	3.00
clust_log_kurtosis	35.30	3.50	0.00	clust_min	35.40	3.60	6.00
avg_neighbor_deg_skew_positive	44.40	1.90	3.00	deg_skew_positive	44.90	1.50	3.00
weight_skew_positive	47.50	0.50	0.00	avg_deg_conn_skew_positive	48.00	0.60	0.00
clust_skew_positive	49.70	0.40	0.00	weight_const	50.80	0.30	0.00
weight_max	51.10	0.40	0.00	core_skew_positive	51.40	0.30	0.00
clust_const	51.60	0.40	0.00	core_const	51.70	0.20	0.00
deg_const	54.50	0.10	0.00	avg_deg_conn_const	54.80	0.10	0.00
avg_neighbor_deg_const	55.10	0.10	0.00	disconnected	55.40	0.10	0.00

Table 4 The variable importance of each feature averaged over all the heuristic-specific random forest models, showing overall importance in predicting heuristic performance for any given instance, as described in Section 6.1. Here, MDGr is the mean decrease in Gini rank, MDG is the mean decrease in Gini, and pct is the percentage of random forest models for which this feature was in the top 10 most important variables. Variables are sorted in increasing order by the mean decrease in Gini rank, with ties broken by the mean decrease in Gini.

Heuristic	R^2	Figures	Heuristic	R^2	Figures	Heuristic	R^2	Figures
ALK98	0.49	Figure 1	BEA98SA	0.41	Figure 2	BEA98TS	0.75	Figure 3
BUR02	0.61	Figure 4	DUA05	0.55	Figure 5	FES02G	0.78	Figure 6
FES02GP	0.76	Figure 7	FES02GV	0.60	Figure 8	FES02GVP	0.68	Figure 9
FES02V	0.37	Figure 10	FES02VP	0.28	Figure 11	GLO10	0.60	Figure 12
GLO98	0.58	Figure 13	HAS00GA	0.54	Figure 14	HAS00TS	0.43	Figure 15
KAT00	0.48	Figure 16	KAT01	0.39	Figure 17	LAG09CE	0.20	Figure 18
LAG09HCE	0.49	Figure 19	LOD99	0.50	Figure 20	LU10	0.71	Figure 21
MER02GR	0.75	Figure 22	MER02GRK	0.50	Figure 23	MER02LS1	0.54	Figure 24
MER02LSK	0.58	Figure 25	MER04	0.30	Figure 26	MER99CR	0.50	Figure 27
MER99LS	0.46	Figure 28	MER99MU	0.24	Figure 29	PAL04MT	0.67	Figure 30
PAL04T1	0.75	Figure 31	PAL04T2	0.29	Figure 32	PAL04T3	0.32	Figure 33
PAL04T4	0.75	Figure 34	PAL04T5	0.68	Figure 35	PAL06	0.61	Figure 36
PAR08	0.65	Figure 37						

Table 1: The R^2 and figure number for each heuristic's CART model predicting instance-specific performance, as described in Section 5.1.