Submodular Maximization with a Knapsack Constraint

Alina Ene
Boston University

Joint work with
Huy Nguyen (Northeastern University)

BOSTON UNIVERSITY

Northeastern University
College of Computer and Information Science

Input:

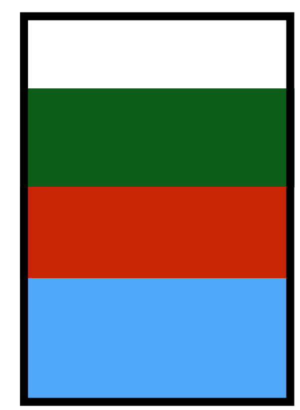
Items $V = \{e_1, e_2, \ldots, e_n\}$ with costs c_{e_i}

$f : 2^V \rightarrow \mathbb{R}_+$ submodular and monotone

Goal:

$$\max_{S \subseteq V} \; f(S)$$

$$\text{s.t.} \;\; c(S) \leq 1$$



NP-hard, admits a tight $1 - \frac{1}{e}$ approx

Approach 1: Guess most valuable items + Density Greedy

Approach 1: Guess most valuable items + Density Greedy

Order the optimal solution OPT

$$o_1, o_2, o_3, \ldots, o_k$$

$$o_i = \operatorname*{argmax}_{o \in \mathrm{OPT} \setminus \{o_1, \ldots, o_{i-1}\}} f(\{o_1, \ldots, o_{i-1}\} \cup \{o\}) - f(\{o_1, \ldots, o_{i-1}\})$$

Approach 1: Guess most valuable items + Density Greedy

Order the optimal solution OPT

$$o_1, o_2, o_3, \ldots, o_k$$

$$o_i = \underset{o \in \mathrm{OPT} \setminus \{o_1, \ldots, o_{i-1}\}}{\operatorname{argmax}} f(\{o_1, \ldots, o_{i-1}\} \cup \{o\}) - f(\{o_1, \ldots, o_{i-1}\})$$

[Sviridenko '04]

Guess the first 3 items o_1, o_2, o_3

Fill up remaining budget using Density Greedy

Approach 1: Guess most valuable items + Density Greedy

Order the optimal solution OPT

$$o_1, o_2, o_3, \ldots, o_k$$

$$o_i = \operatorname*{argmax}_{o \in \mathrm{OPT} \setminus \{o_1, \ldots, o_{i-1}\}} f(\{o_1, \ldots, o_{i-1}\} \cup \{o\}) - f(\{o_1, \ldots, o_{i-1}\})$$

[Sviridenko '04]

Guess the first 3 items o_1, o_2, o_3

Fill up remaining budget using Density Greedy

$1 - \frac{1}{e}$ approx in time $O(n^5)$ (time = function evals)

$1 - \frac{1}{e} - \epsilon$ approx in time $\tilde{O}(n^4)$ via lazy Density Greedy

Approach 2: Guess (approximate) marginal values

OPT_1

$o_1, o_2, \ldots, o_t$

large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

Guess (approximately)

- The marginal value of every item in OPT_1
- The total marginal value of OPT_2

Greedily pick items to meet these marginal values

Approach 2: Guess (approximate) marginal values

[Badanidiyuru, Vondrak '14]

$$\begin{aligned} \max \quad & F(x) \\ \text{s.t.} \quad & \langle c, x \rangle \le 1 \\ & x \in [0,1]^n \end{aligned}$$

Large Items: Pack using Continous Greedy
Small Items: Pack using Continuous Density Greedy

Round the fractional solution without any loss

$1 - \frac{1}{e} - \epsilon$ approx in time $O\left(n^2 \cdot \left(\frac{\log n}{\epsilon}\right)^{\frac{1}{\epsilon^8}}\right)$

[Badanidiyuru, Vondrak '14]

Solve the multilinear relaxation in time $\tilde{O}_\epsilon(n^2)$

Round solution without loss in time $\tilde{O}(n)$

Where does the quadratic running time come from?

The algorithm is actually very efficient

It evaluates F(x) only $\tilde{O}_\epsilon(n)$ times

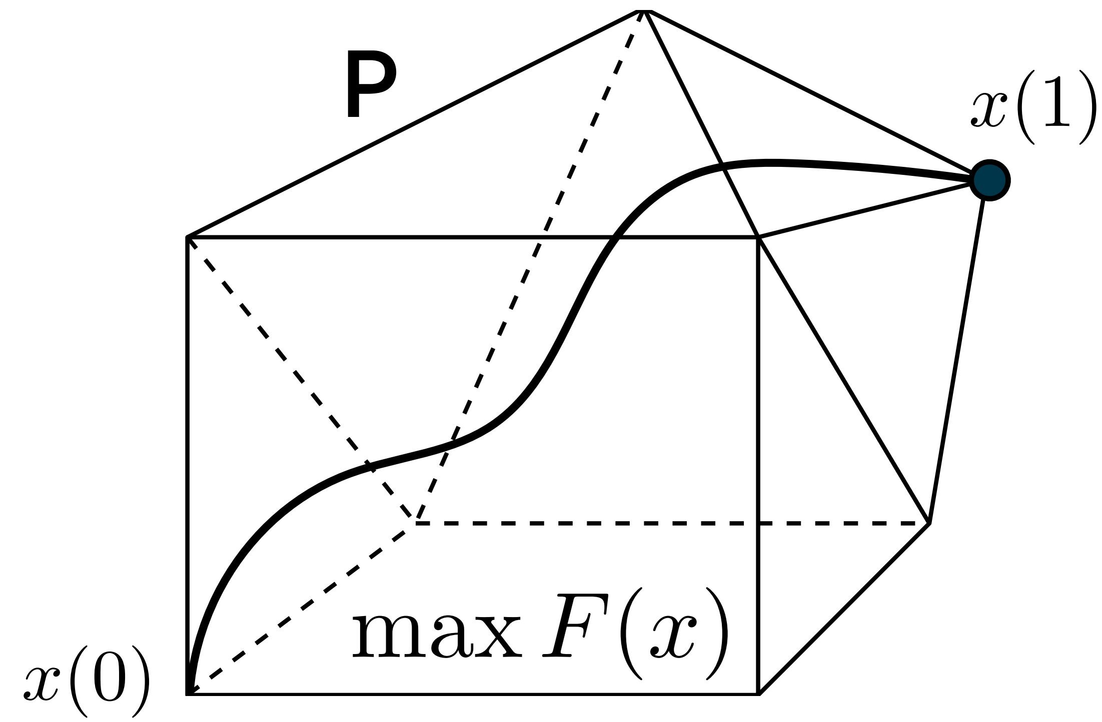
Each F(x) evaluation requires evaluating f O(n) times

Multilinear Relaxation and Rounding Paradigm

Solve $\max\{f(S) : S \in \mathcal{F}\}$

relax ↓ ↑ round

Solve $\max\{F(x) : x \in P\}$



Technical challenge

Develop submodular maximization algorithms that use fewer than n^2 evaluation queries

Conceptual challenge

Avoid generic uses of the multilinear extension

[Badanidiyuru, Vondrak '14]

Solve the multilinear relaxation in time $\tilde{O}_\epsilon(n^2)$

Round solution without loss in time $\tilde{O}(n)$

Overall running time: $O\left(n^2\left(\frac{\log n}{\epsilon}\right)^{\frac{1}{\epsilon^8}}\right)$

Our contributions:

Algorithm for knapsack with $\tilde{O}_\epsilon(n)$ evaluations

Ensure x only has $O_\epsilon(1)$ fractional entries

Each F(x) evaluation takes $O_\epsilon(1)$ value queries

Overall running time: $\left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon^4}\right)} n\log^2 n$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

For $p = 1, 2, \ldots, 1/\epsilon$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t$

Guess $v_{p,i} \approx F(x \vee \mathbf{1}_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon \mathbf{1}_{e_{p,i}}$

OPT_1 OPT_2

$o_1, o_2, \ldots, o_t$ $o_{t+1}, o_{t+2}, \ldots, o_k$

$t = \frac{1}{\epsilon^3}$ large value items

small value items

For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t$

Guess $v_{p,i} \approx F(x \vee \mathbf{1}_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon \mathbf{1}_{e_{p,i}}$

Guess $w_p \approx F(x \vee \mathbf{1}_{\mathrm{OPT}_2}) - F(x)$

Use Density Greedy to integrally select a set S_p

with total gain $F(x \vee \mathbf{1}_{S_p}) - F(x) \approx \epsilon w_p$

Update $x \leftarrow x \vee \mathbf{1}_{S_p}$

The algorithm constructs a solution x with:

$$F(x) \geq \left(1 - \frac{1}{e} - \epsilon\right) f(\mathrm{OPT})$$

The algorithm constructs a solution x with:

$$F(x) \geq \left(1 - \frac{1}{e} - \epsilon\right) f(\mathrm{OPT})$$

$\frac{1}{\epsilon^4}$ fractional entries that can be charged to OPT_1

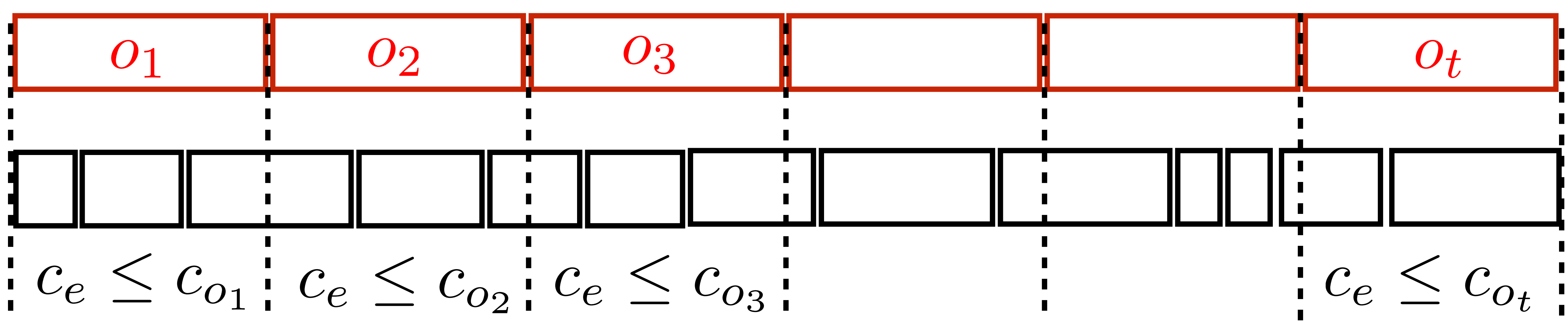
For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t \;\; (t = 1/\epsilon^3)$

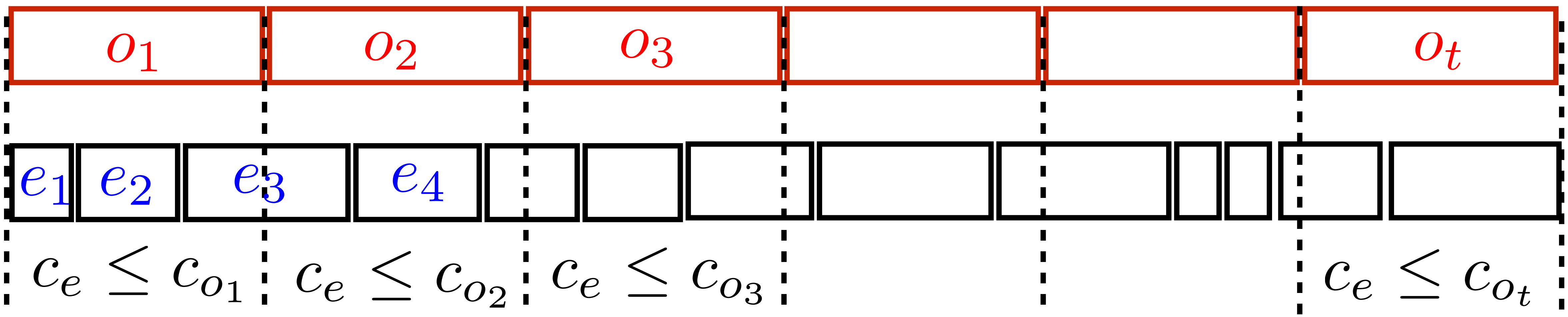
Guess $v_{p,i} \approx F(x \vee \mathbf{1}_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon \mathbf{1}_{e_{p,i}}$

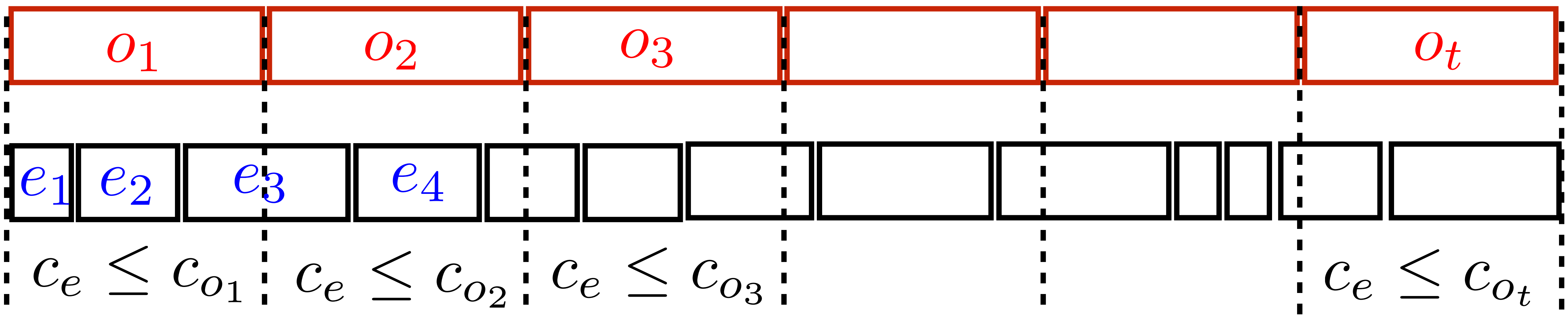


Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$



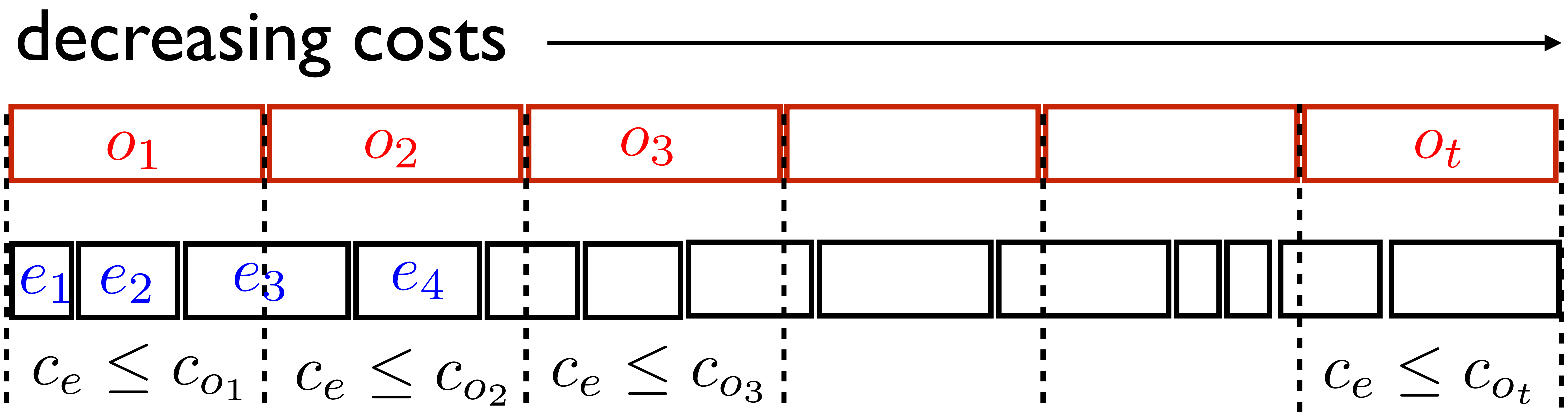
Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



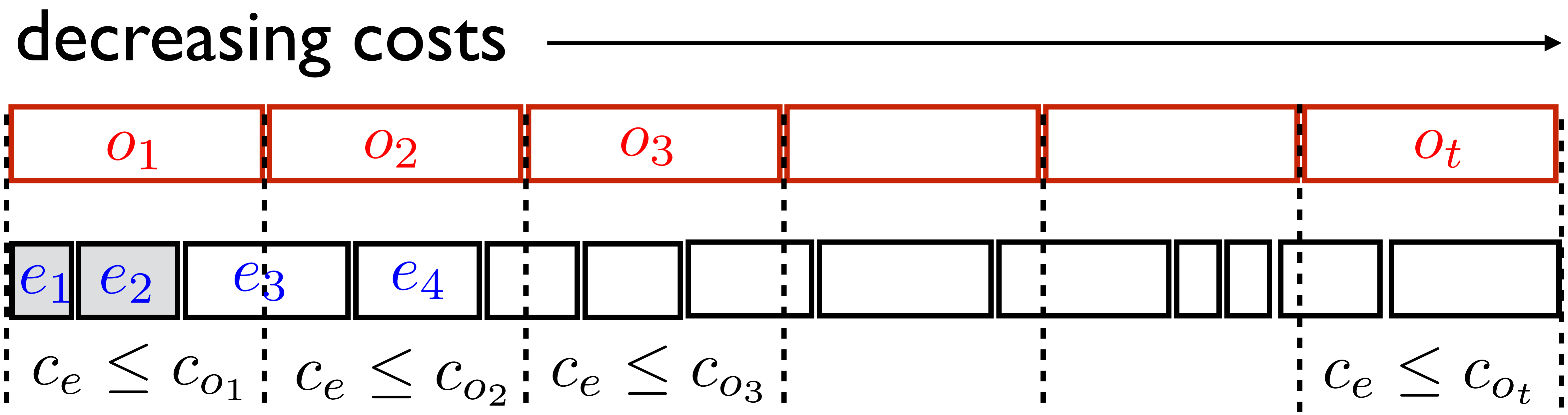
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



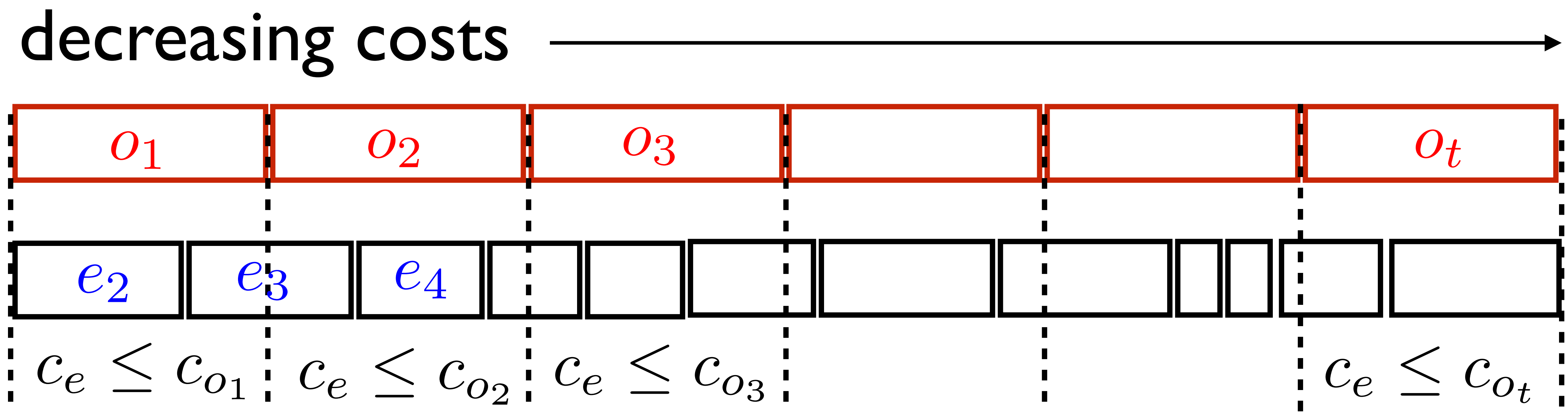
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



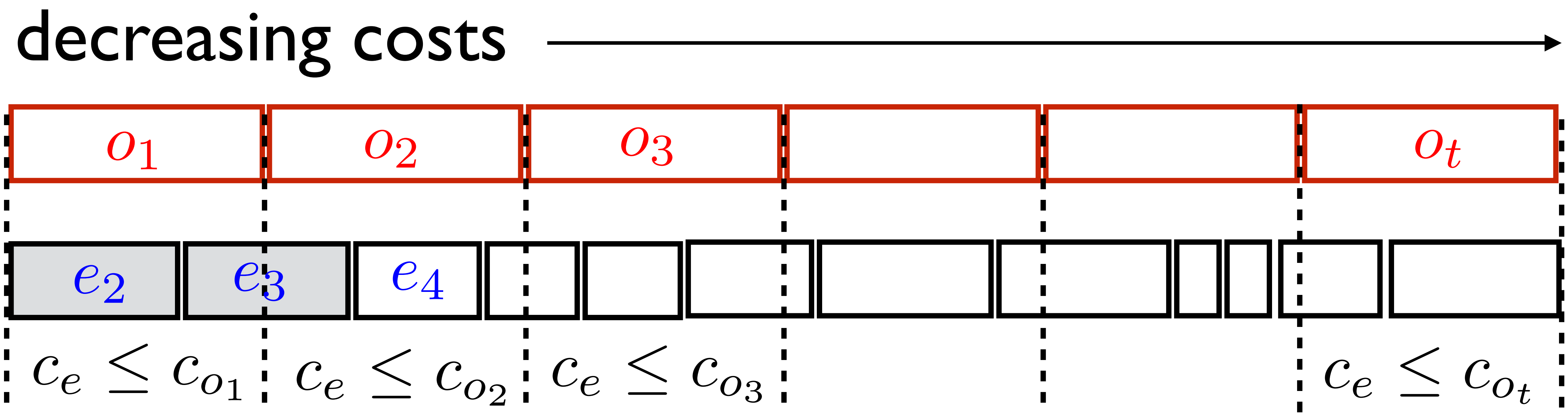
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



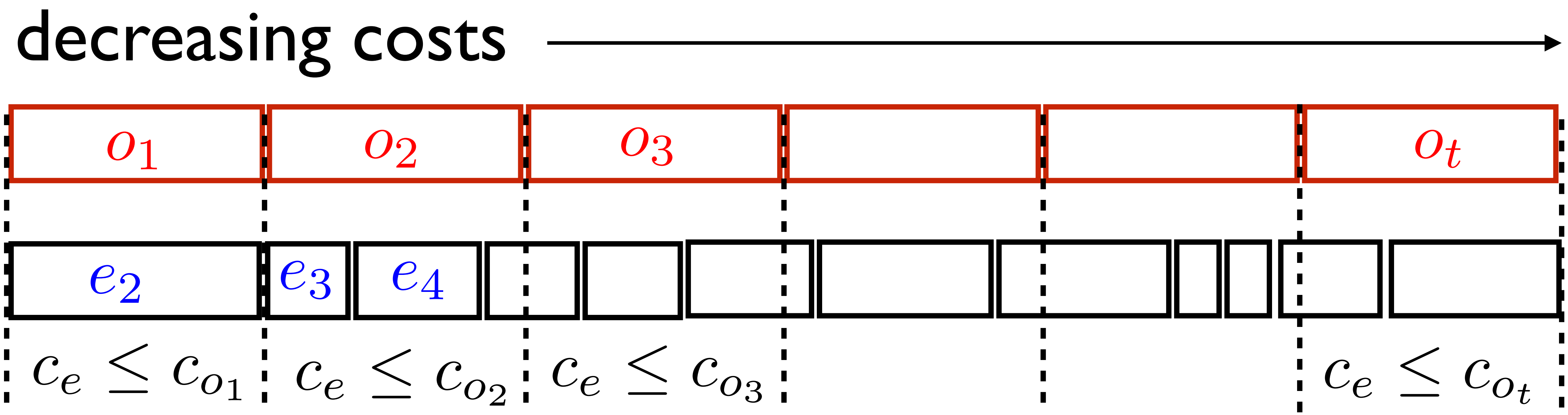
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



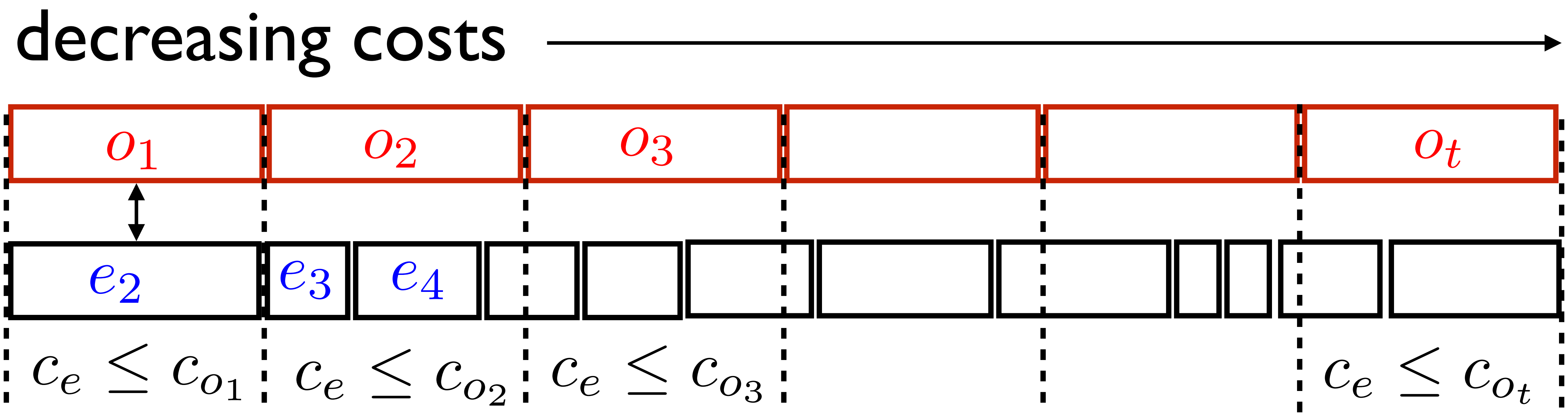
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



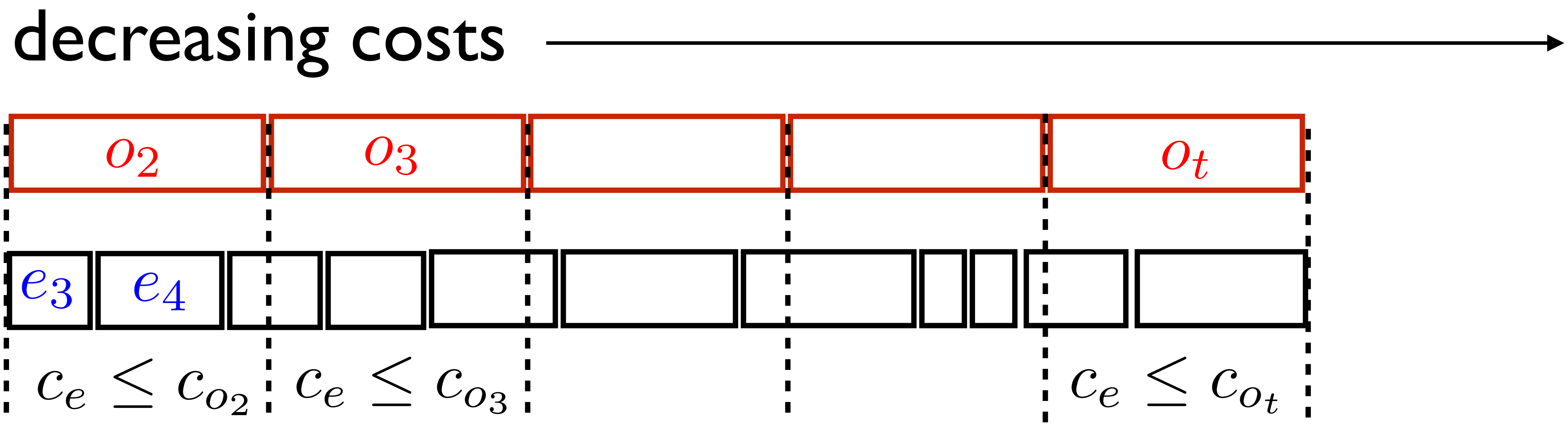
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



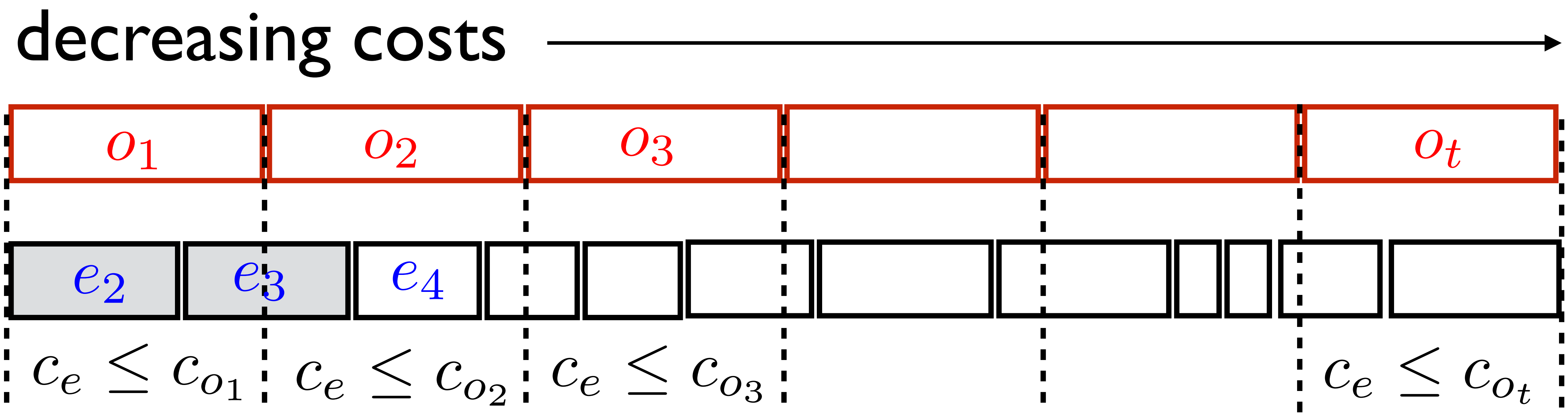
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



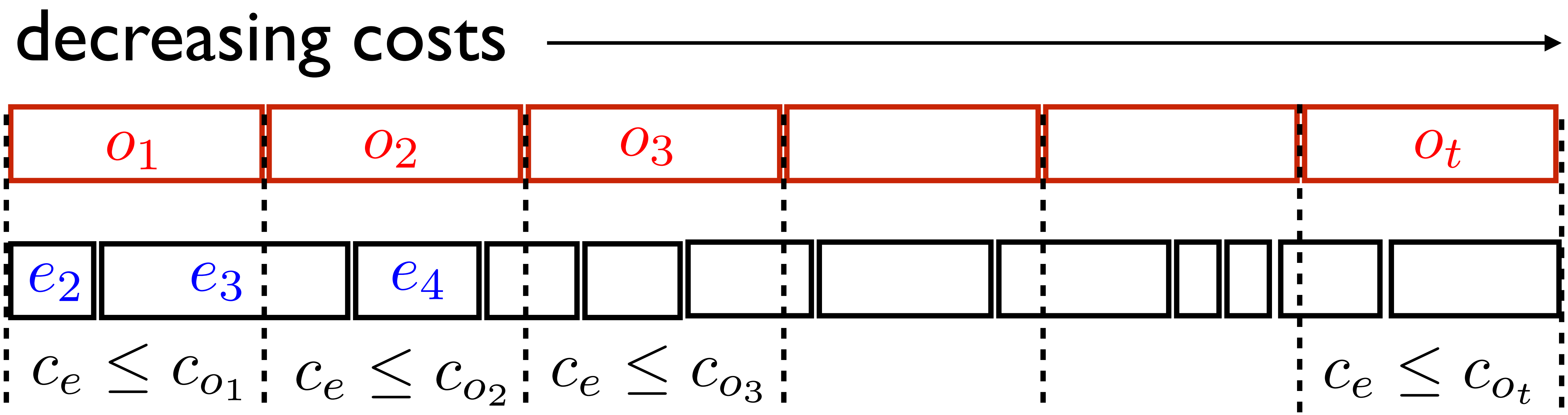
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



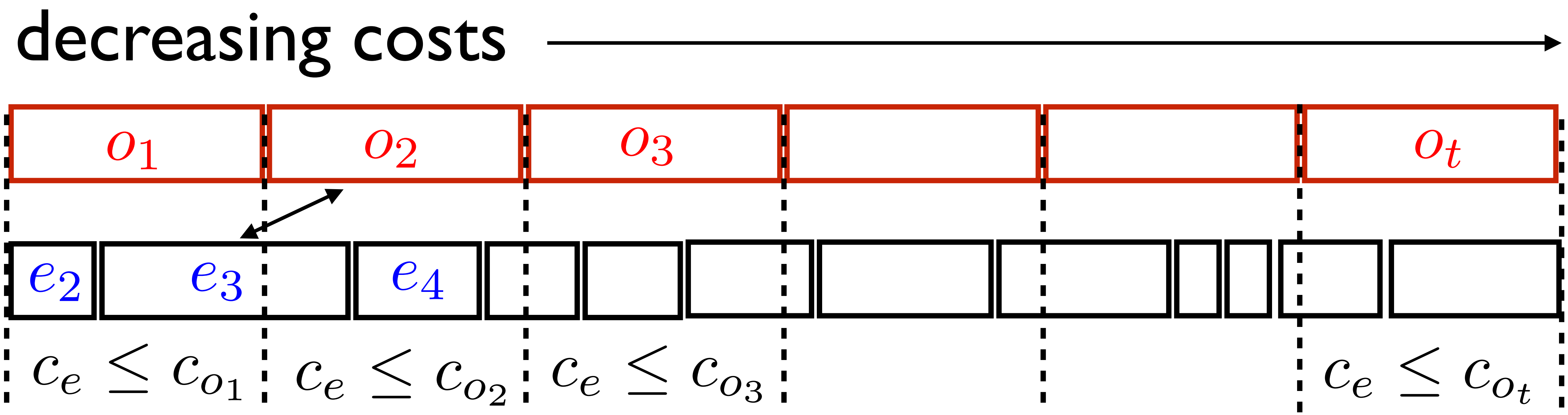
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



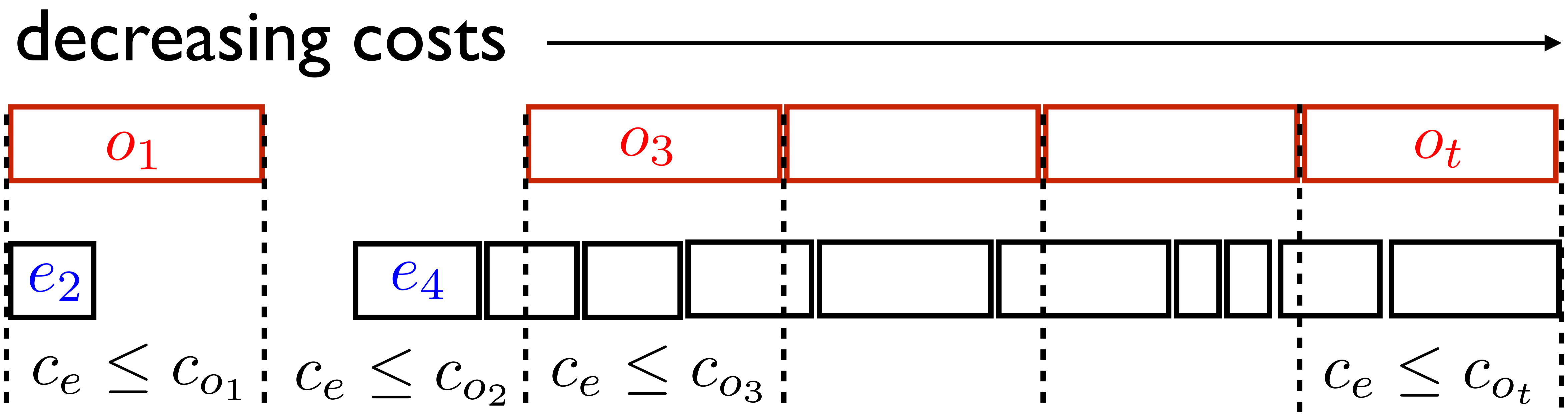
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



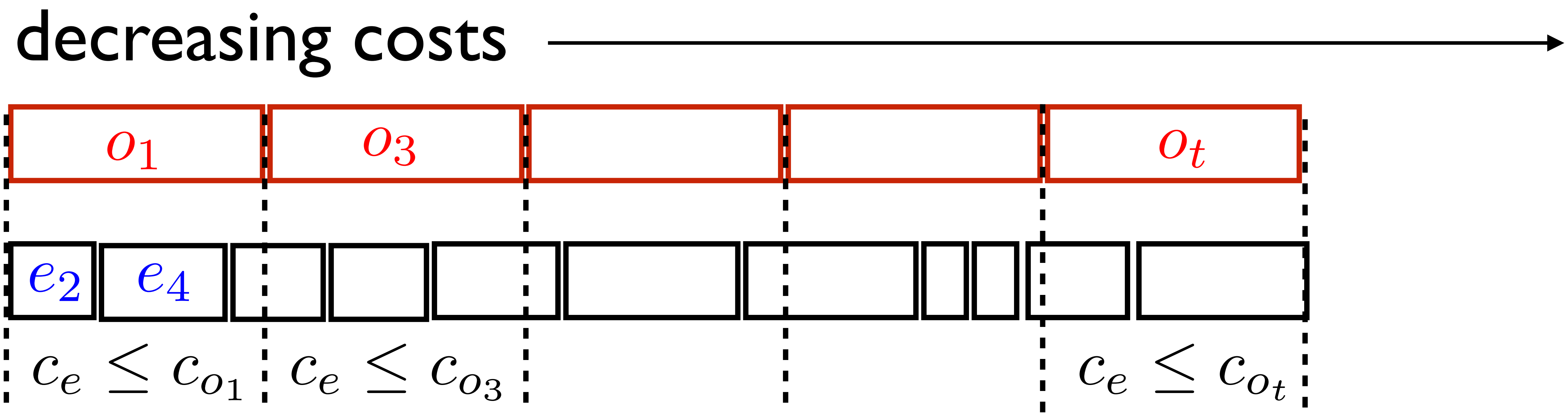
Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

Round: preserve value, cost $\leq c(\mathrm{OPT}_1)$

Intuition: similar to a partition matroid



Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(\mathbf{1}_{e_1} - \mathbf{1}_{e_2}) \text{ where } \mathbf{Ex}[\delta] = 0$$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t$

Guess $v_{p,i} \approx F(x \vee \mathbf{1}_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon \mathbf{1}_{e_{p,i}}$

Guess $w_p \approx F(x \vee \mathbf{1}_{\mathrm{OPT}_2}) - F(x)$

Use Density Greedy to integrally select a set S_p

with total gain $F(x \vee \mathbf{1}_{S_p}) - F(x) \approx \epsilon w_p$

Update $x \leftarrow x \vee \mathbf{1}_{S_p}$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

For $p = 1, 2, \ldots, 1/\epsilon$

Need to ensure that Density Greedy picks items with cost at most $1 - c(\mathrm{OPT}_1)$

Guess $w_p \approx F(x \vee \mathbf{1}_{\mathrm{OPT}_2}) - F(x)$

Use Density Greedy to integrally select a set S_p with total gain $F(x \vee \mathbf{1}_{S_p}) - F(x) \approx \epsilon w_p$

Update $x \leftarrow x \vee \mathbf{1}_{S_p}$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

Can assume that each item in OPT_2 has small cost

$$c_o \leq \epsilon^2(1 - c(\mathrm{OPT}_1)) \quad \forall o \in \mathrm{OPT}_2$$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

Can assume that each item in OPT_2 has small cost

$$c_o \leq \epsilon^2(1 - c(\mathrm{OPT}_1)) \quad \forall o \in \mathrm{OPT}_2$$

Each item $o \in \mathrm{OPT}_2$ satisfies

$$f(\mathrm{OPT}_1 \cup \{o\}) - f(\mathrm{OPT}_1) \leq \epsilon^3 f(\mathrm{OPT})$$

There are at most $1/\epsilon^2$ items $o \in \mathrm{OPT}_2$ with cost

$$c_o > \epsilon^2(1 - c(\mathrm{OPT}_1))$$

Discard all of them and lose only $\epsilon f(\mathrm{OPT})$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

Can assume that each item in OPT_2 has small cost

$$c_o \leq \epsilon^2(1 - c(\mathrm{OPT}_1)) \quad \forall o \in \mathrm{OPT}_2$$

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

Can assume that each item in OPT_2 has small cost

$$c_o \leq \epsilon^2(1 - c(\mathrm{OPT}_1)) \quad \forall o \in \mathrm{OPT}_2$$

If each item in OPT_2 also has small marginal gain $F(x \vee \mathbf{1}_o) - F(x)$ then Density Greedy items will fit

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

Can assume that each item in OPT_2 has small cost

$$c_o \leq \epsilon^2(1 - c(\mathrm{OPT}_1)) \quad \forall o \in \mathrm{OPT}_2$$

If each item in OPT_2 also has small marginal gain $F(x \vee \mathbf{1}_o) - F(x)$ then Density Greedy items will fit

Items in OPT_2 only have small gain on top of OPT_1

Marginal gain on top of x could be large

OPT_1

$o_1, o_2, \ldots, o_t$

$t = \frac{1}{\epsilon^3}$ large value items

OPT_2

$o_{t+1}, o_{t+2}, \ldots, o_k$

small value items

Can assume that each item in OPT_2 has small cost

$$c_o \leq \epsilon^2(1 - c(\mathrm{OPT}_1)) \quad \forall o \in \mathrm{OPT}_2$$

If each item in OPT_2 also has small marginal gain $F(x \vee \mathbf{1}_o) - F(x)$ then Density Greedy items will fit

Items in OPT_2 only have small gain on top of OPT_1

Marginal gain on top of x could be large

Can fix this issue using more guessing, similarly to OPT_1

Summary and Open Questions

Overall running time: $\left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon^4}\right)} n \log^2 n$

Improve the dependency on ϵ? (FPTAS for linear fns)

Faster running times for matroid constraints?

[BV '14] $O\left(\frac{rn}{\epsilon^4} \log^2\left(\frac{r}{\epsilon}\right)\right)$ time

Partition matroid is a natural (and challenging) case