Budget-Feasible Mechanism Design for Non-Monotone Submodular Objectives

Georgios Amanatidis

University of Essex

Pieter Kleer

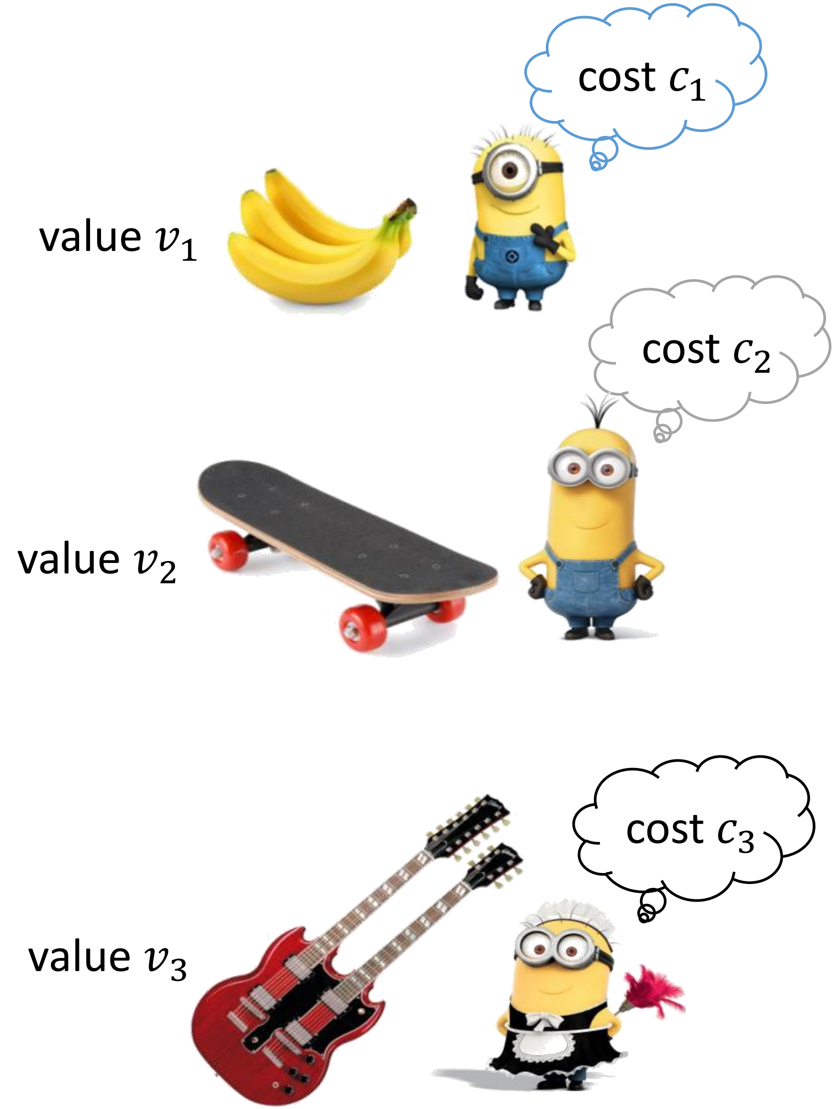
Max-Planck-Institut für Informatik

Guido Schäfer

Centrum Wiskunde & Informatica

the setting

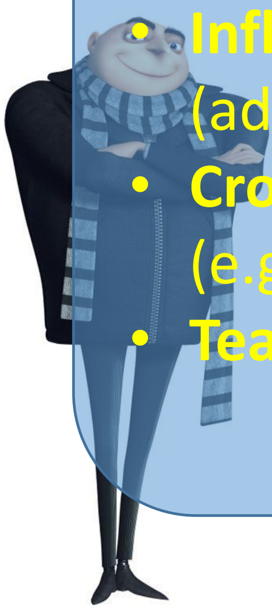
Buyer with budget B
and valuation function v



the setting

Models applications like:

- **Influence maximization** (advertisement on social networks)
- **Crowdsourcing platforms** (e.g., Amazon Mechanical Turk, ClickWorker)
- **Team formation**



Buyer with budget B and valuation function v

value v_1

cost c_1

value v_2

cost c_2

value v_3

cost c_3

the setting

Buyer with budget B

value v_1  cost c_1

value v_2  cost c_2

value v_3  cost c_3

the setting

- Set of items $A = \{1, 2, \ldots, n\}$.
- Each item i comes with a cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \rightarrow \mathbb{R}$.

the setting

Buyer with budget B
and an additive
valuation function

value v_1  cost c_1

value v_2  cost c_2

Total value = $v_2 + v_3$

value v_3  cost c_3

the setting

- Set of items $A = \{1, 2, \ldots, n\}$.
- Each item i comes with a cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.

the setting

- Set of items $A = \{1, 2, \dots, n\}$.
- Each item i comes with a cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.
- When v is additive:
  - Objective: Select a set S that maximizes $v(S) = \sum_{i \in S} v_i$ subject to the constraint $\sum_{i \in S} c_i \leq B$.
  - This is just Knapsack!

the setting

- Knapsack is an NP-hard problem.

Reminder:

ALG is a ρ-approximation algorithm if $\rho \cdot v\big(ALG(I)\big) \geq v\big(OPT(I)\big)$ for all I.

- However, we can approximate the optimal solution within $1 + \epsilon$ in polynomial time.

- Straightforward 2-approximation algorithm:
  - Sort all items from higher to lower density (*value / cost*);
  - Greedily build a feasible solution S w.r.t. this ordering;
  - Return the best among S and the item of highest value.

the setting

Buyer with budget B
and a submodular
valuation function

value v_1  cost c_1

value v_2  cost c_2

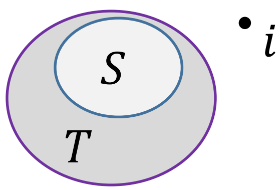
Total value ≤ $v_2 + v_3$

value v_3  cost c_3

the setting

- Set of items $A = \{1, 2, \dots, n\}$.
- Each item i comes with a cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.
- Typically, v is submodular:
  - $v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)$ for any $S \subseteq T$ and $i \notin T$

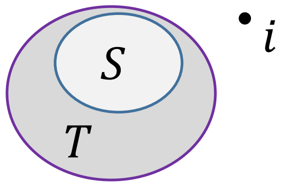
i's marginal contribution decreases as the set grows



the setting

- Set of items $A = \{1, 2, \dots, n\}$.
- Each item i comes with a cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \rightarrow \mathbb{R}$.
- Typically, v is submodular:
  - $v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)$ for any $S \subseteq T$ and $i \notin T$

i's *marginal* contribution decreases as the set grows

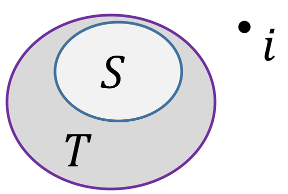


  - Select a set S that maximizes $v(S)$ subject to $\sum_{i \in S} c_i \leq B$.

the setting

- Set of items $A = \{1, 2, \ldots, n\}$.
- Each item i comes with a cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.
- Typically, v is submodular:
  - $v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)$ for any $S \subseteq T$ and $i \notin T$

i's *marginal* contribution decreases as the set grows



  - Select a set S that maximizes $v(S)$ subject to $\sum_{i \in S} c_i \leq B$.
  - Known e-approximation algorithm

the setting

- This is still an NP-hard problem.

- Approximating the optimal solution within $\frac{e}{e-1}$ in polynomial time is the best one could hope for.

- Straightforward 3-approximation algorithm for *monotone* submodular objectives:
  - Sort all items from higher to lower *marginal* density;
  - Greedily build a feasible solution S w.r.t. this ordering;
  - Return the best among S and the item of highest value.

the setting

Buyer with budget B and a submodular valuation function

cost c_1

value v_1

cost c_2

value v_2

cost c_3

value v_3

the setting

- Set of agents $A = \{1, 2, \dots, n\}$.
- Each agent i comes with a *private* cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.

the setting

- Set of agents $A = \{1, 2, \ldots, n\}$.
- Each agent i comes with a *private* cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \rightarrow \mathbb{R}$.
  Here v is *general submodular.*

the setting

- Set of agents $A = \{1, 2, \dots, n\}$.
- Each agent i comes with a *private* cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.
  Here v is *general submodular.*

Reminder:

A function $v: 2^A \to \mathbb{R}$ is *submodular* if for any $S \subseteq T$ and $i \notin T$:

$$v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)\ .$$

the setting

- Set of agents $A = \{1, 2, \ldots, n\}$.
- Each agent i comes with a *private* cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.
  Here v is *general submodular.*

Find a set S that maximizes $v(S)$, subject to $\sum_{i \in S} c_i \leq B$.

the setting

Buyer with budget B and a submodular valuation function

cost c_1

value v_1

cost c_2

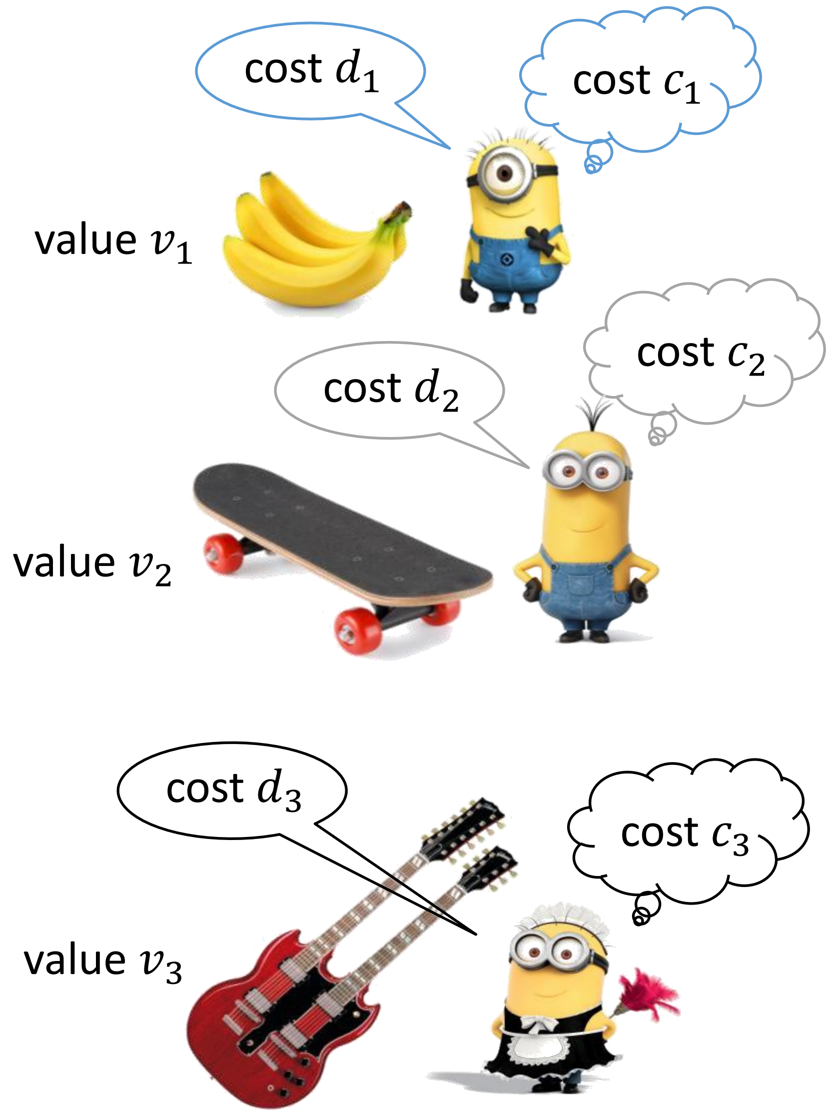
value v_2

cost c_3

value v_3

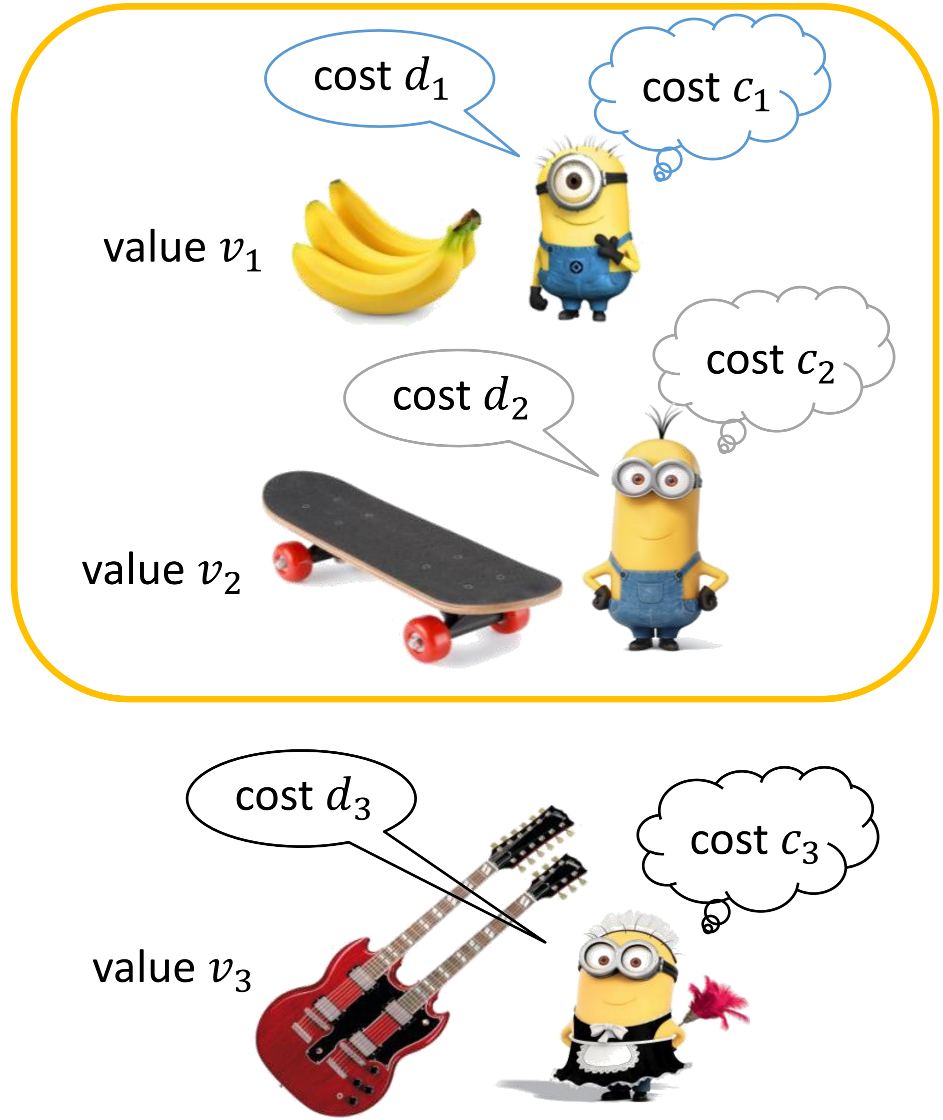
the setting

Buyer with budget B
and a submodular
valuation function

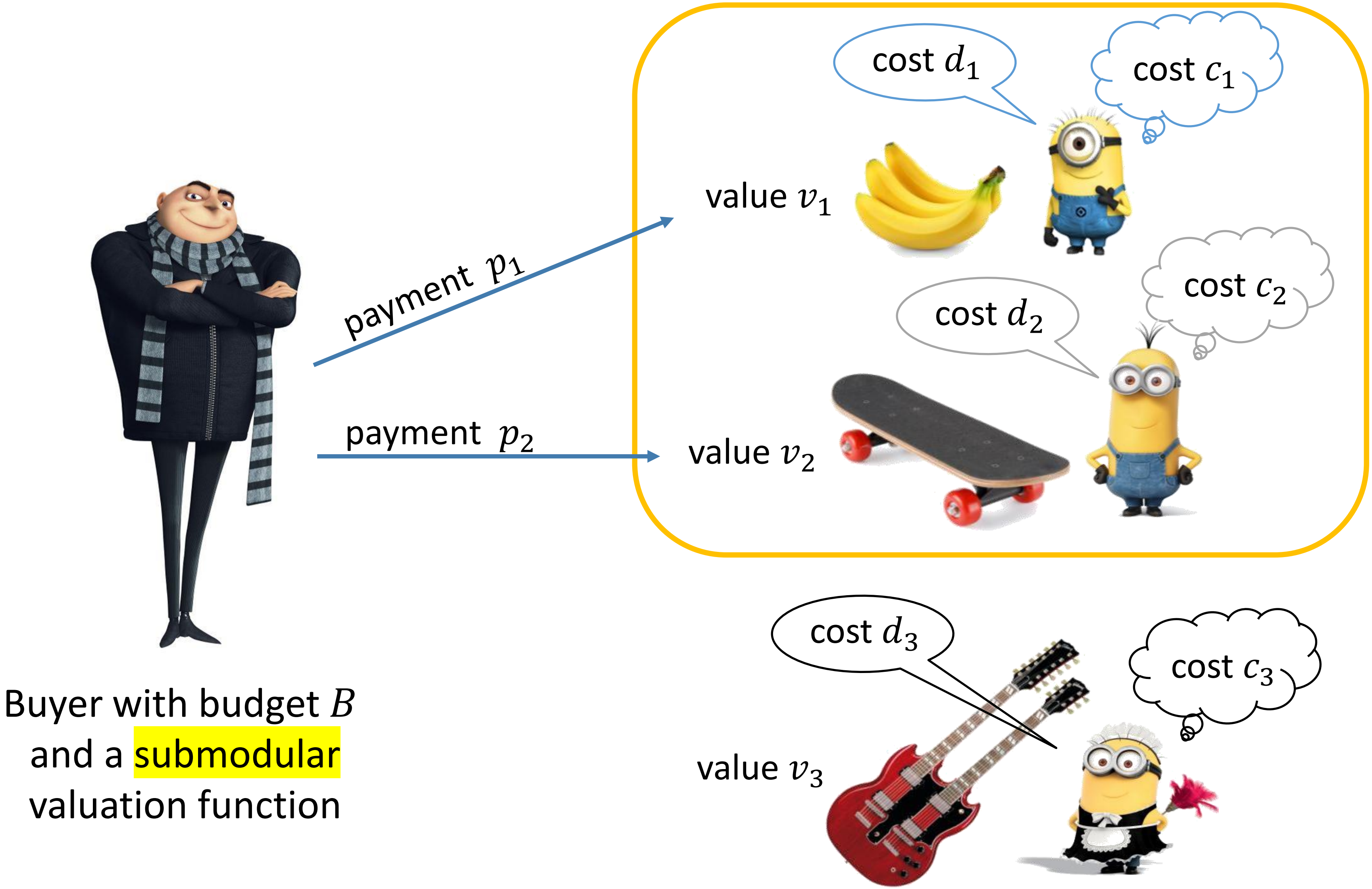


the setting

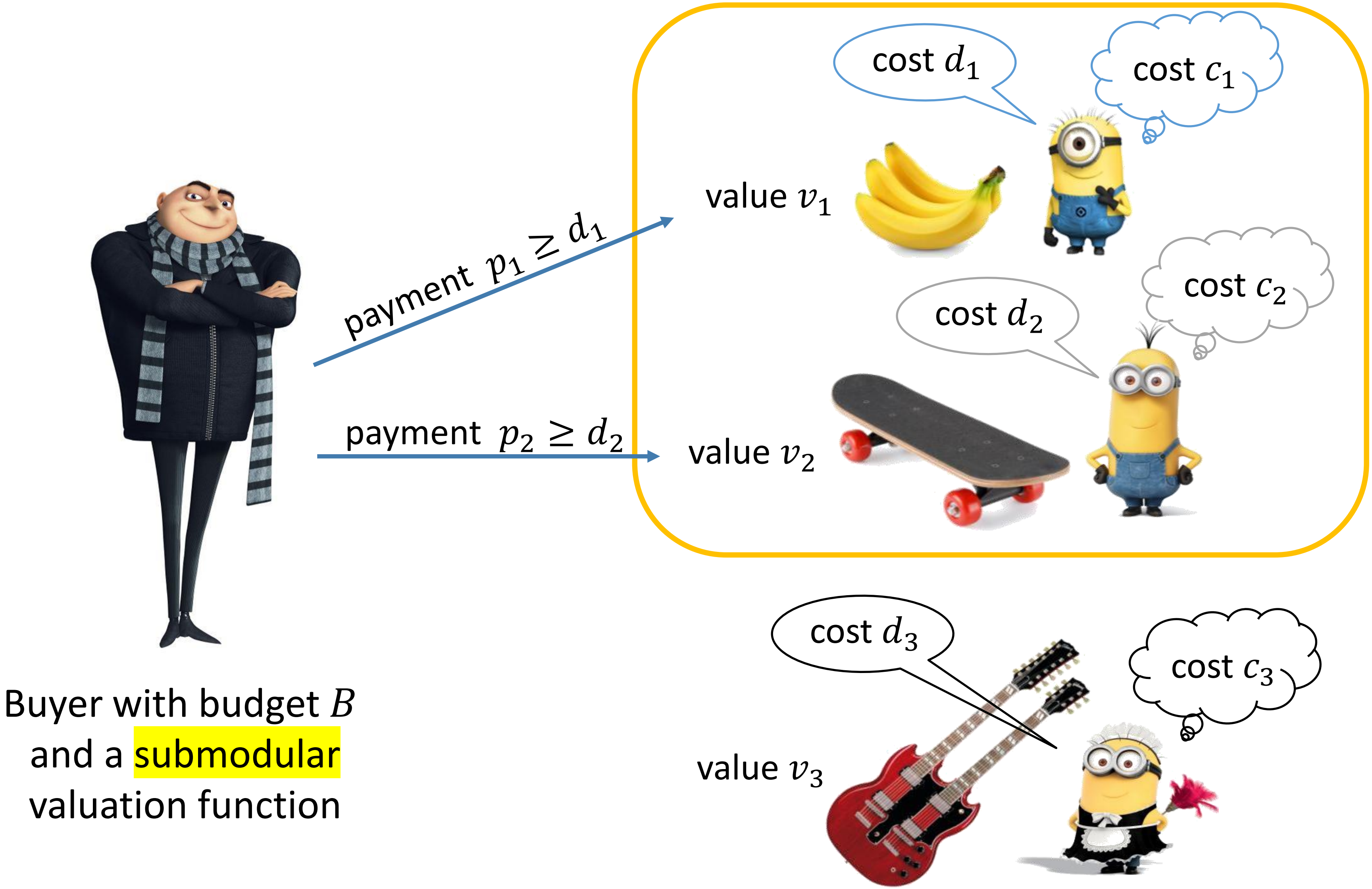
Buyer with budget B and a submodular valuation function



the setting


cost d_1
cost c_1
value v_1
payment p_1
cost d_2
cost c_2
payment p_2
value v_2
cost d_3
cost c_3
value v_3
Buyer with budget B and a submodular valuation function

the setting


cost d_1
cost c_1
value v_1
payment $p_1 \geq d_1$
cost d_2
cost c_2
payment $p_2 \geq d_2$
value v_2
cost d_3
cost c_3
value v_3
Buyer with budget B
and a submodular
valuation function

the setting

- Can we ensure that the agents report the c_is?

the setting

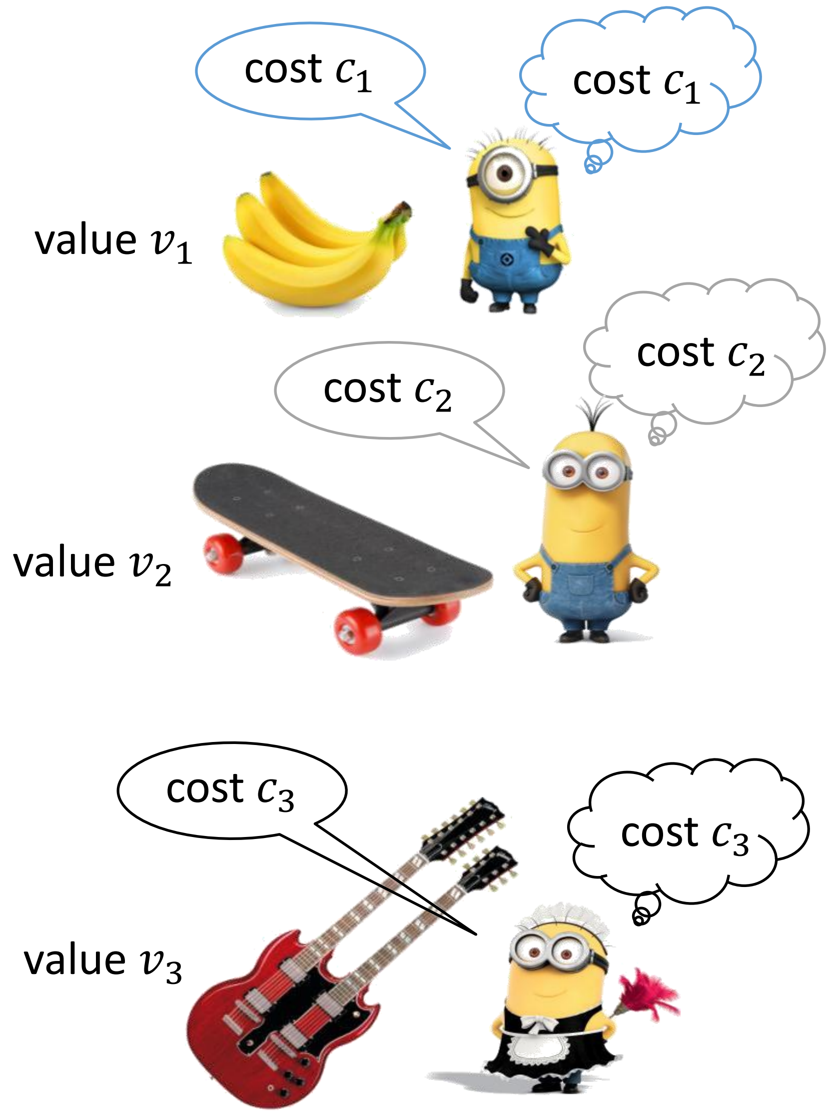
- Can we ensure that the agents report the c_is?
- A **truthful** mechanism is an algorithm that uses payments to ensure that *no agent has an incentive to lie*.

the setting

- Can we ensure that the agents report the c_is?
- A **truthful** mechanism is an algorithm that uses payments to ensure that *no agent has an incentive to lie*.
- In settings like this one, there is a unique payment scheme that works, given that our solution is monotone (Myerson)

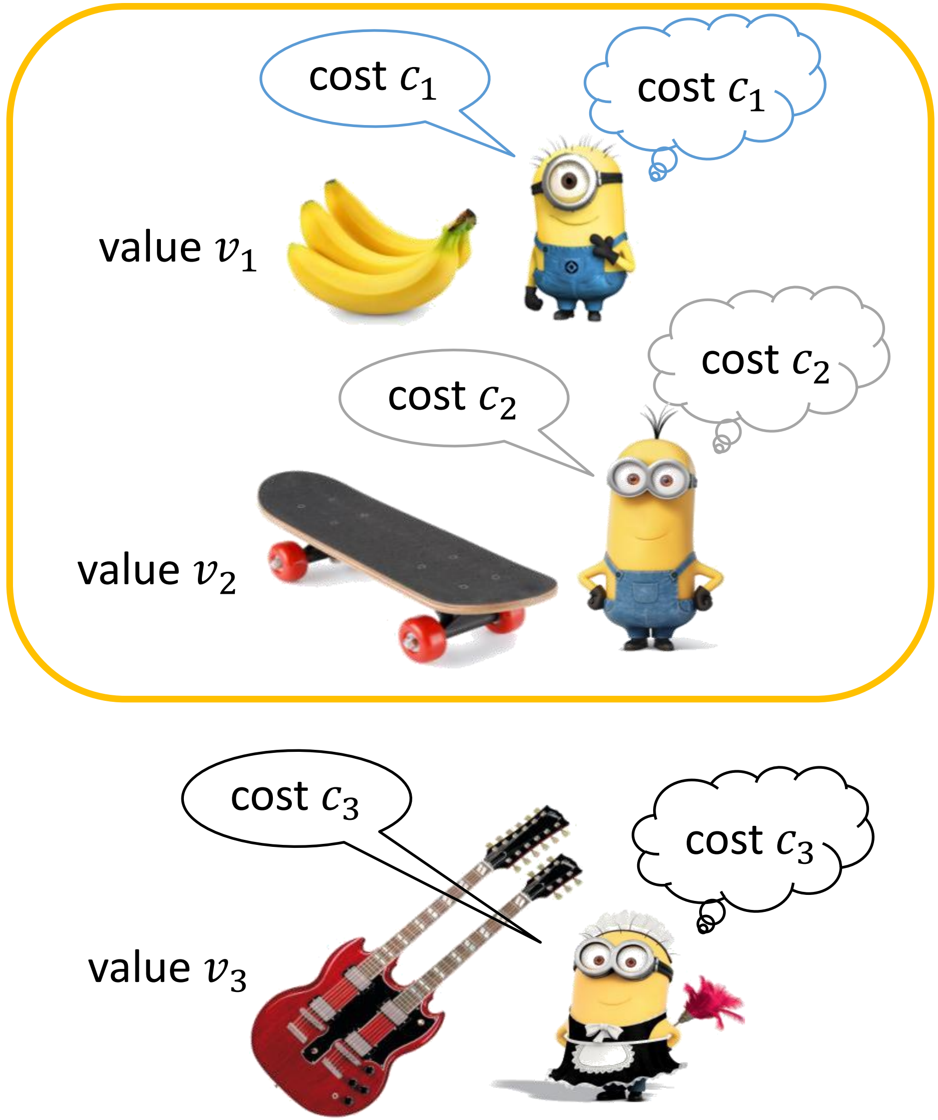
the setting

Buyer with budget B
and a submodular
valuation function

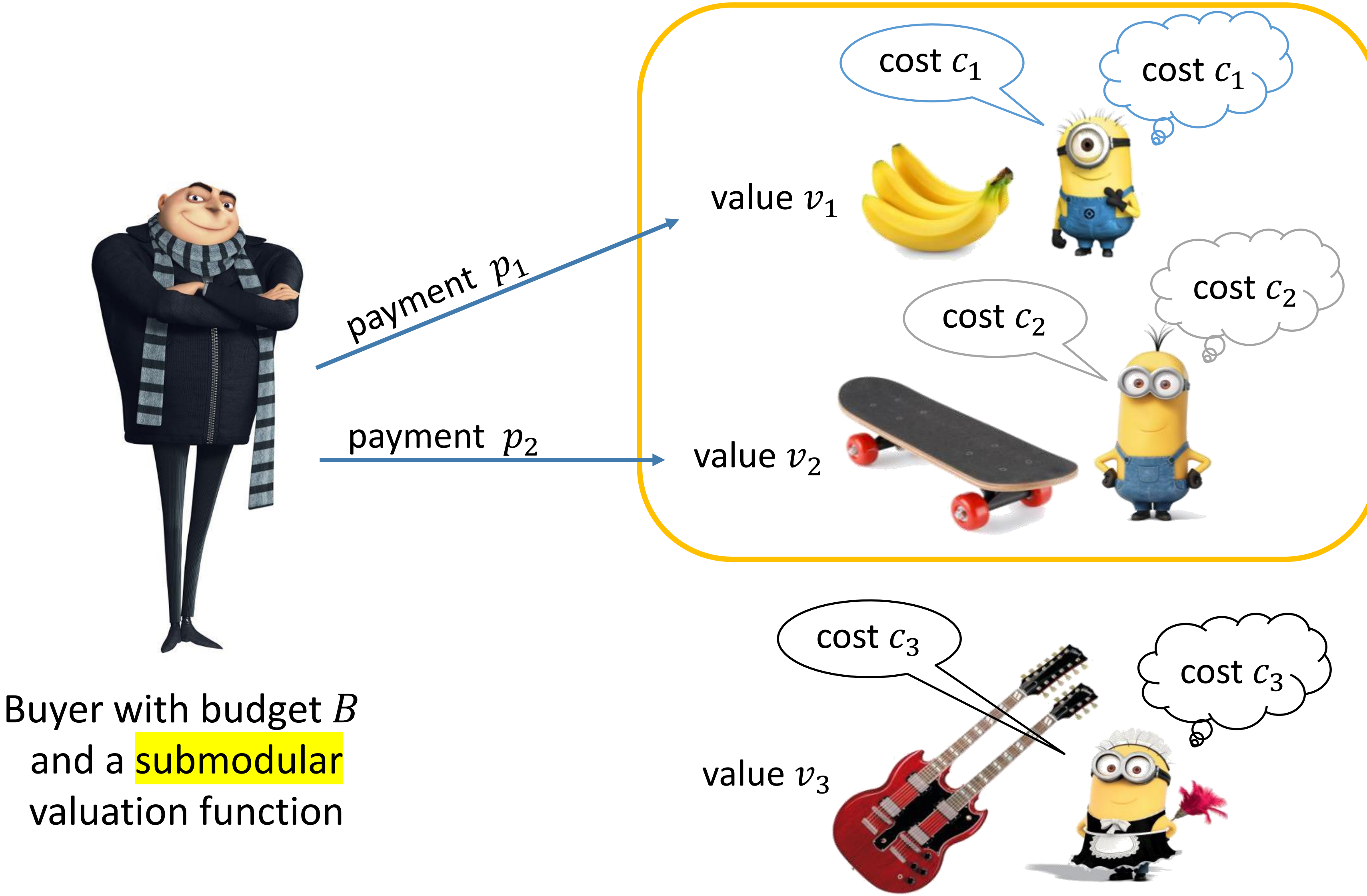


the setting

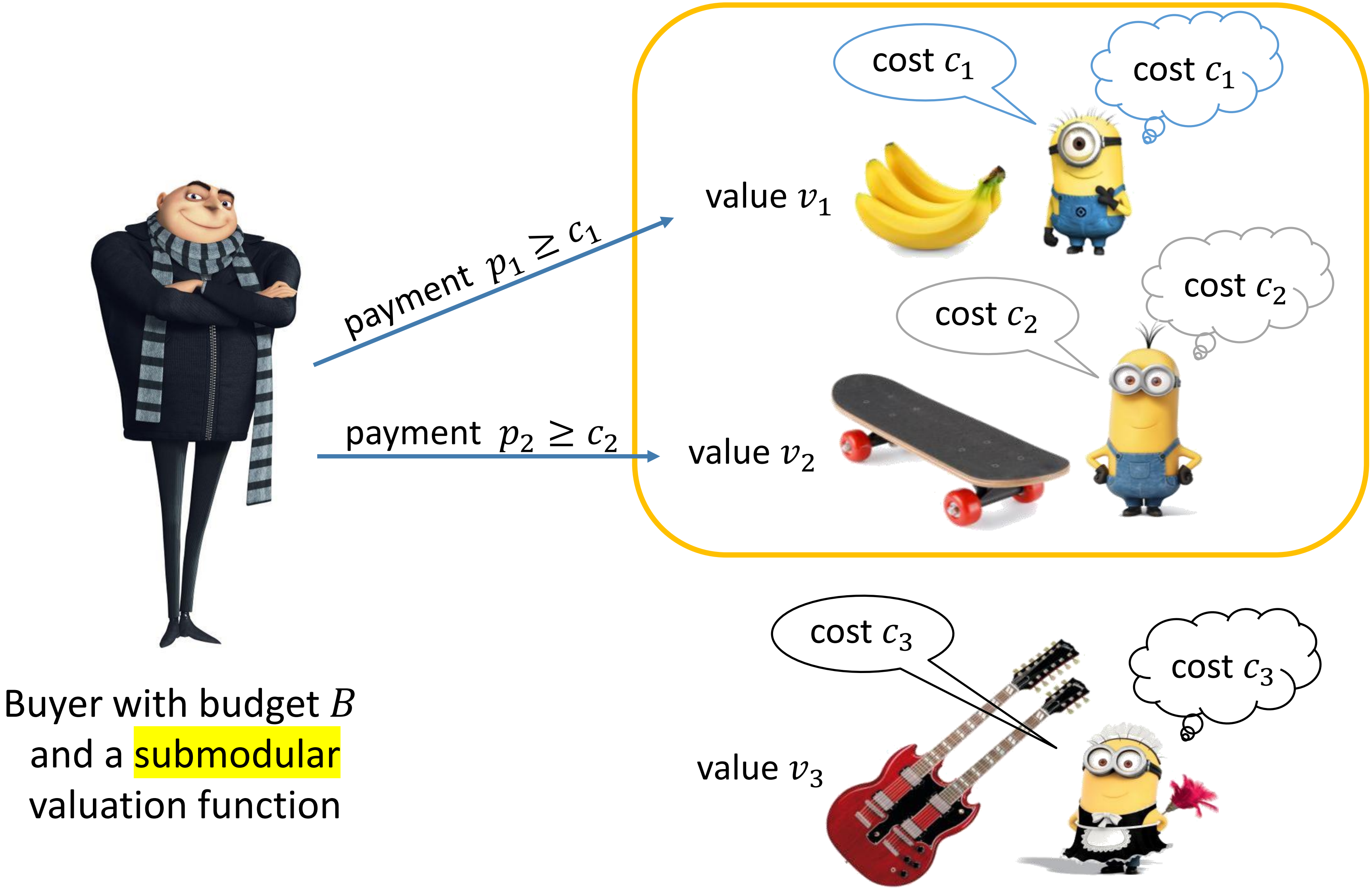
Buyer with budget B and a submodular valuation function



the setting


cost c_1
cost c_1
value v_1
payment p_1
cost c_2
cost c_2
payment p_2
value v_2
cost c_3
cost c_3
value v_3
Buyer with budget B
and a submodular
valuation function

the setting



cost c_1
cost c_1
value v_1
payment $p_1 \geq c_1$
cost c_2
cost c_2
payment $p_2 \geq c_2$
value v_2
cost c_3
cost c_3
value v_3
Buyer with budget B and a submodular valuation function

the setting

- Set of agents $A = \{1, 2, \ldots, n\}$.
- Each agent i comes with a *private* cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.
  Here v is *general submodular.*

Find a set S that maximizes $v(S)$, subject to $\sum_{i \in S} c_i \leq B$.

Design **truthful** mechanisms with strong approximation guarantees.

the setting

- Set of agents $A = \{1, 2, \ldots, n\}$.
- Each agent i comes with a *private* cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \rightarrow \mathbb{R}$.
  Here v is *general submodular.*

Find a set S that maximizes $v(S)$, subject to $\sum_{i \in S} c_i \leq B$.

Design **truthful**, **budget-feasible** mechanisms with strong approximation guarantees.

the setting

- Set of agents $A = \{1, 2, \ldots, n\}$.
- Each agent i comes with a *private* cost c_i.
- Buyer with a budget B and a valuation function $v: 2^A \to \mathbb{R}$.
  Here v is *general submodular.*

Find a set S that maximizes $v(S)$, subject to ~~$\sum_{i \in S} c_i \leq B$~~.

$\sum_{i \in S} p_i \leq B$

Design **truthful**, **budget-feasible** mechanisms with strong approximation guarantees.

related work

- Initiated by [Singer ’10]
- Additive and monotone submodular objectives
  [Singer ’10], [Chen, Gravin, Lu ’11], [Badanidiyuru, Kleinberg, Singer ’12], [A., Birmpas, Markakis ’16], [Leonardi, Monaco, Sankowski, Zhang ’17], [Jalaly, Tardos ’18], [Gravin ’19]
- Subadditive, XOS, and symmetric submodular objectives
  [Dobzinski, Singer, Papadimitriou ’11], [Bei, Chen, Gravin, Lu ’12], [A., Birmpas, Markakis ’17]
- For general submodular objectives an exponential-time 768-approximation mechanism is implied by [Bei et al. ’12]

budget-feasible mechanism design

- *Single-parameter* mechanism design problem.
- Suffices to find monotone algorithms. (Myerson's lemma)

Myerson's lemma

- Designing of truthful mechanisms (almost) the same as constructing monotone allocation rules.

- We say that an outcome rule f is *monotone*, if

$$i \in f(b_i, b_{-i}) \Rightarrow i \in f(b_i', b_{-i}) \text{ for } b_i' \leq b_i$$

i's bid

Everyone else's bid (vector)

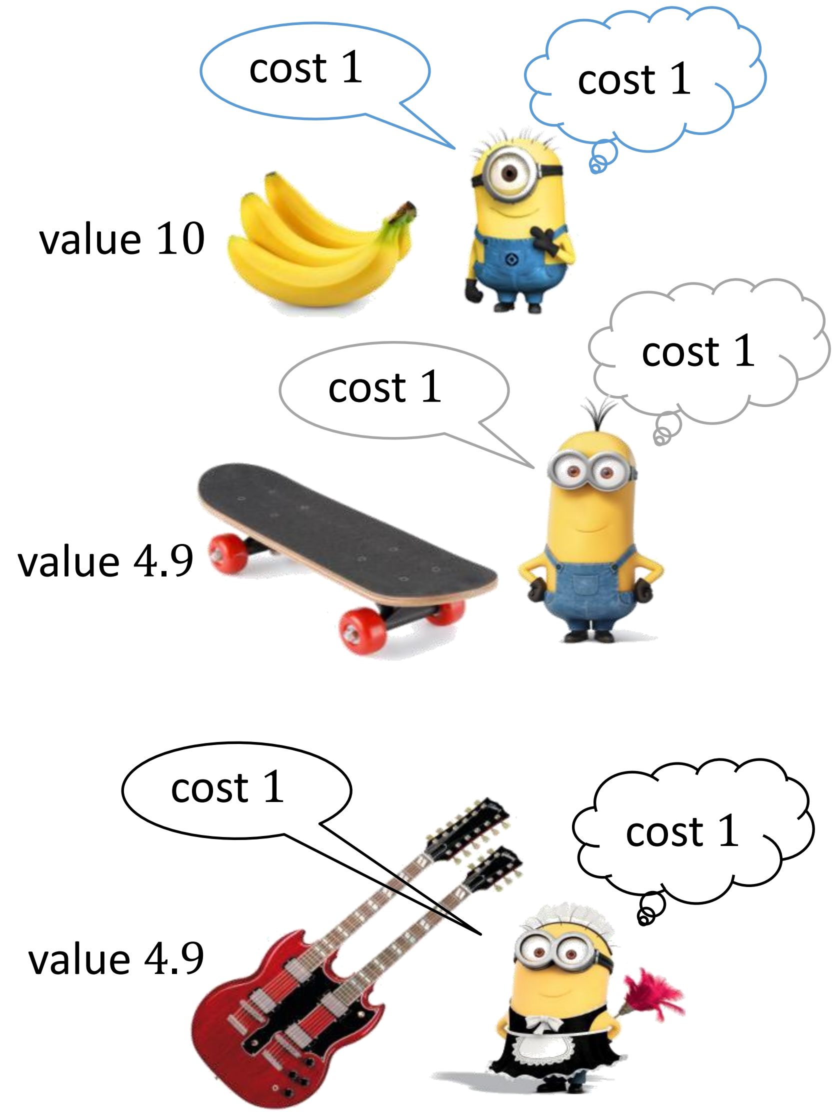
Lemma: Given a monotone algorithm f, there is a unique payment scheme p such that (f, p) is a truthful and individually rational mechanism.

budget-feasible mechanism design

- *Single-parameter* mechanism design problem.
- Suffices to find monotone algorithms. (Myerson's lemma)
- Presence of budget makes the problem very challenging.
- Even exponential truthful mechanisms are not obvious.

lower bound

Buyer with budget
$B = 3$
and an additive
valuation function



lower bound

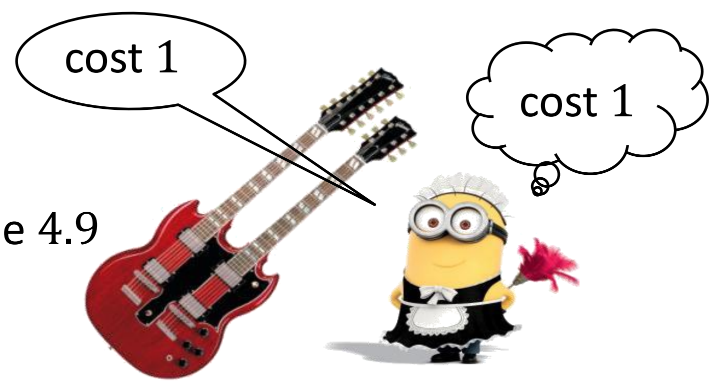
value of optimal solution $= 19.8$

Buyer with budget
$B = 3$
and an additive
valuation function

value 10

value 4.9

value 4.9



lower bound

Must be included to the solution, or else we have an approximation factor > 2.

How much should he get paid?

Buyer with budget
$B = 3$
and an additive
valuation function

cost 1

cost 1

value 10

cost 1

cost 1

value 4.9

cost 1

cost 1

value 4.9

lower bound

Must be included to the solution, or else we have an approximation factor > 2.

How much should he get paid?

Buyer with budget

$B = 3$

and an additive valuation function

suppose $p_1 < 3$

cost 1

cost 1

value 10

cost 1

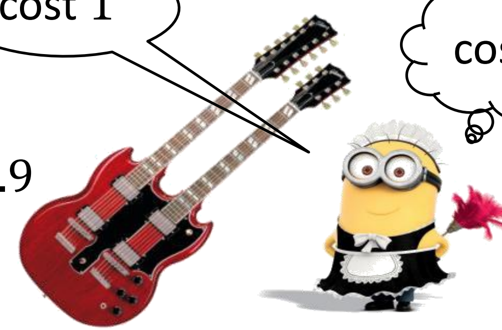
cost 1

value 4.9

cost 1

cost 1

value 4.9

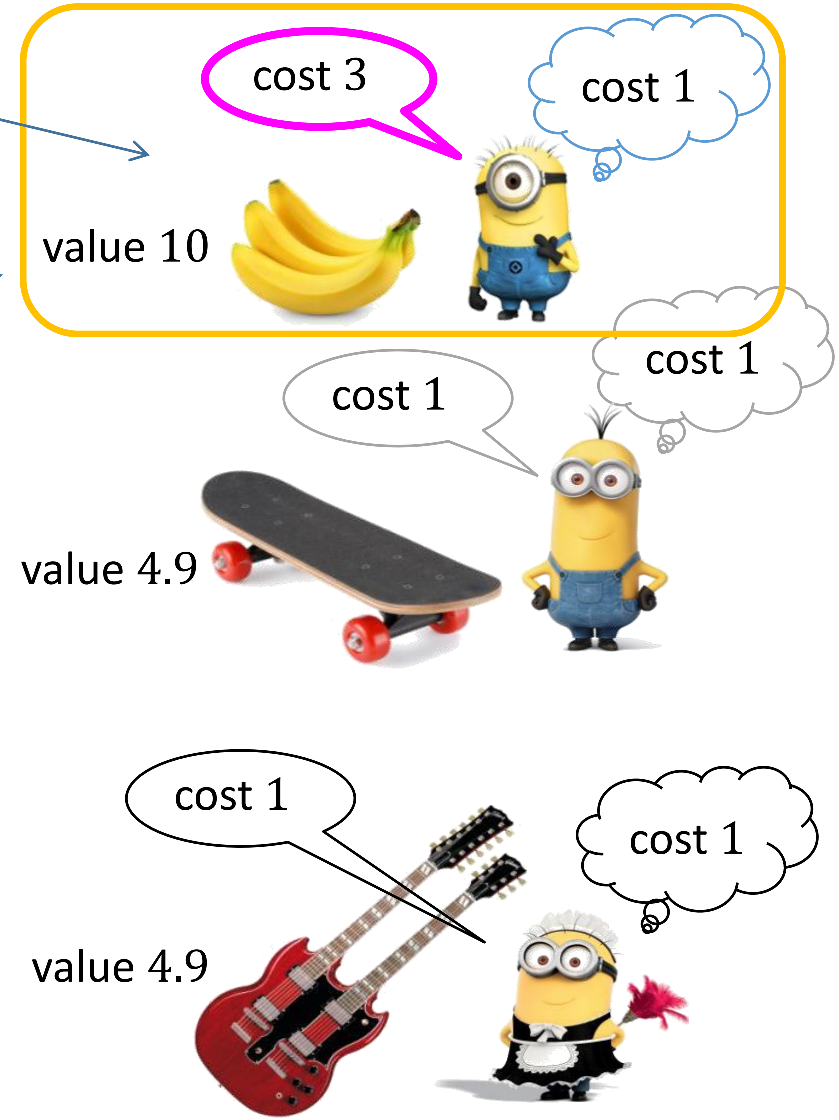


lower bound

Must be included to the solution, or else we have an approximation factor > 2.

How much should he get paid?

suppose $p_1 < 3$



Buyer with budget

$B = 3$

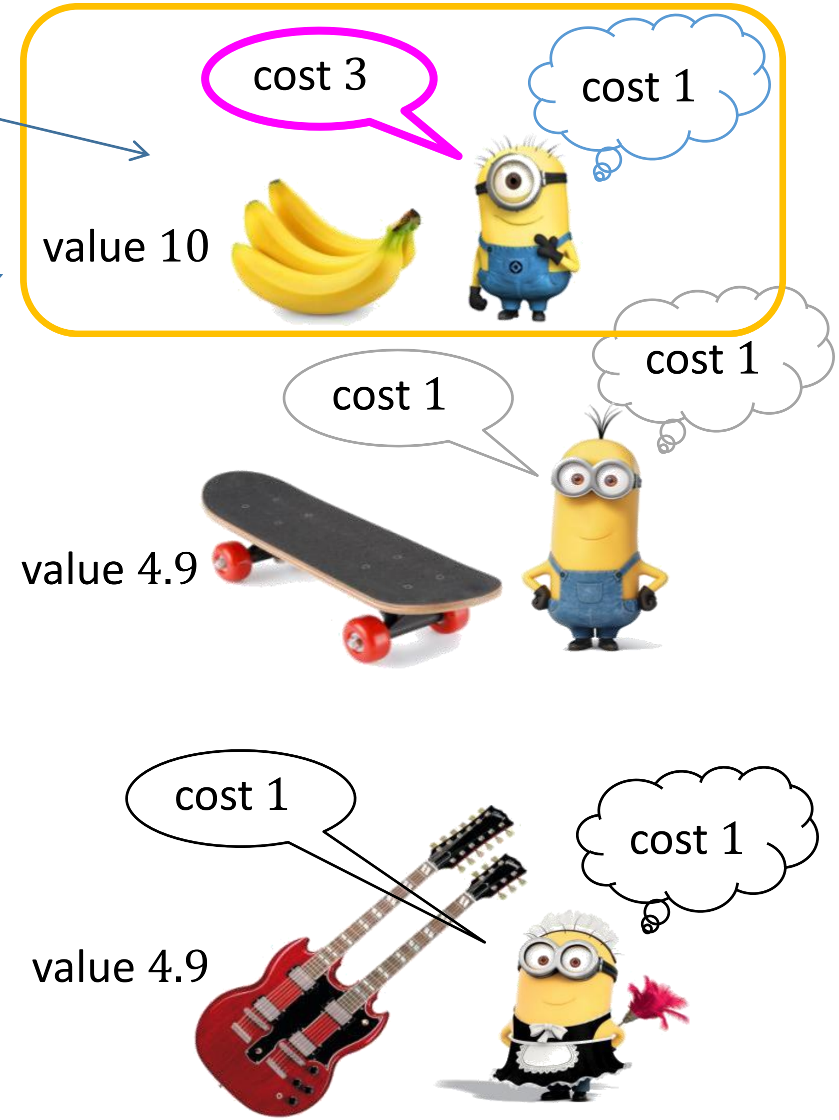
and an additive valuation function

lower bound

Must be included to the solution, or else we have an approximation factor > 2.

How much should he get paid?

must pay $p_1 = 3$



Buyer with budget
$B = 3$
and an additive
valuation function

lower bound

Must be included to the solution, or else we have an approximation factor > 2.

How much should he get paid?

Buyer with budget
$B = 3$
and an additive
valuation function

must pay $p_1 = 3$
even when he tells the truth!

cost 1

cost 1

value 10

cost 1

cost 1

value 4.9

cost 1

cost 1

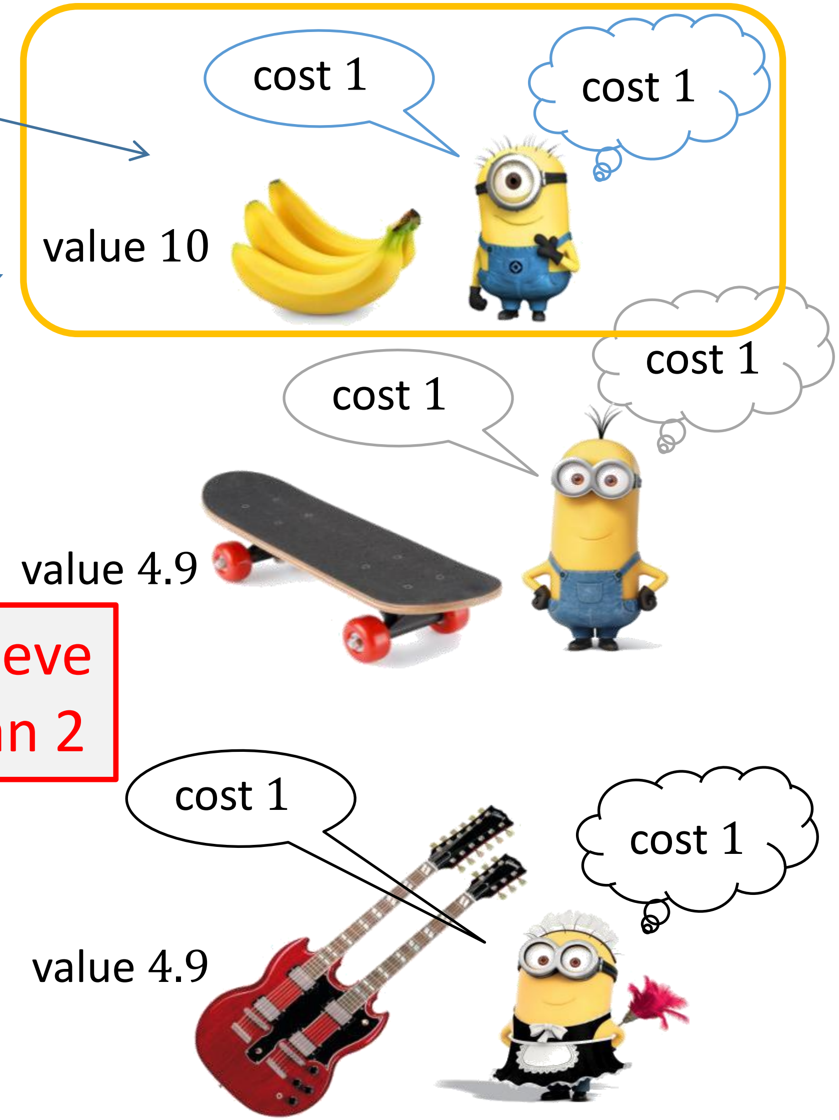
value 4.9

lower bound

Must be included to the solution, or else we have an approximation factor > 2.

How much should he get paid?

must pay $p_1 = 3$ even when he tells the truth!



Impossible to achieve a factor better than 2

Buyer with budget
$B = 3$
and an additive
valuation function

budget-feasible mechanism design

- *Single-parameter* mechanism design problem.
- Suffices to find monotone algorithms. (Myerson's lemma)
- Presence of budget makes the problem very challenging.
- Even exponential truthful mechanisms are not obvious.
- Only widely applicable approach –even for "easier" objectives– is using a very simple greedy subroutine.

related work – general approach

- Existing constant approximation mechanisms boil down to the following:

> Output either the best singleton or a greedy solution.

- Inspired by the 3-approximation algorithm above, the greedy sorts the agents with respect to their marginal value per cost ratio and selects them up to a threshold.

related work – general approach

- Existing constant approximation mechanisms boil down to the following:

> Output either the best singleton or a greedy solution.

- Inspired by the 3-approximation algorithm above, the greedy sorts the agents with respect to their marginal value per cost ratio and selects them up to a threshold.
- For non-monotone submodular objectives, this greedy approach –and many reasonable variants– fails badly.

our results

Main theorem: There is a polynomial-time, universally truthful, budget-feasible O(1)-approximation mechanism for (non-monotone) submodular objectives in the value query model.

our results

Main theorem: There is a polynomial-time, universally truthful, budget-feasible O(1)-approximation mechanism for (non-monotone) submodular objectives in the value query model.

- A function $v: 2^A \to \mathbb{R}$ is submodular if for any $S \subseteq T$ and $i \notin T$:
  $$v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)\ .$$

our results

Main theorem: There is a polynomial-time, universally truthful, budget-feasible O(1)-approximation mechanism for (non-monotone) submodular objectives in the value query model.

- A function $v: 2^A \to \mathbb{R}$ is submodular if for any $S \subseteq T$ and $i \notin T$:
$$v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)\,.$$

- In the value query model, we assume oracle access to v via value queries, i.e., we assume the existence of a polynomial time value oracle that returns $v(S)$ when given as input a set S.

our results

Main theorem: There is a polynomial-time, universally truthful, budget-feasible O(1)-approximation mechanism for (non-monotone) submodular objectives in the value query model.

- A function $v: 2^A \to \mathbb{R}$ is submodular if for any $S \subseteq T$ and $i \notin T$:

$$v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T)\,.$$

- In the value query model, we assume oracle access to v via value queries, i.e., we assume the existence of a polynomial time value oracle that returns $v(S)$ when given as input a set S.

- A randomized mechanism is universally truthful if it is a probability distribution over deterministic truthful mechanisms.

our results

Main theorem: There is a polynomial-time, universally truthful, budget-feasible O(1)-approximation mechanism for (non-monotone) submodular objectives in the value query model.

- The above result can be extended to the online (secretary) setting where the agents arrive in a uniformly random order.

our results

Main theorem: There is a polynomial-time, universally truthful, budget-feasible O(1)-approximation mechanism for (non-monotone) submodular objectives in the value query model.

- The above result can be extended to the online (secretary) setting where the agents arrive in a uniformly random order.
- It can be also be generalized to the setting where the feasible sets satisfy combinatorial constraints.

our results

Main theorem: There is a polynomial-time, universally truthful, budget-feasible O(1)-approximation mechanism for (non-monotone) submodular objectives in the value query model.

- The above result can be extended to the online (secretary) setting where the agents arrive in a uniformly random order.
- It can be also be generalized to the setting where the feasible sets satisfy combinatorial constraints.
- For the broader class of general XOS objectives, exponentially many queries are needed for any non-trivial approximation.

the mechanism

Submodular Mechanism$(A, v, \mathbf{c}, B)$

With probability $p = 1/5$:

  return $i^* \in \arg\max_{i \in A} v(i)$

With probability $1 - p$:

  Put each agent in either A_1 or A_2 independently at random w.p. $\frac{1}{2}$

  $x \approx v(\text{OPT}(A_1))$

  $S_1 = S_2 = \emptyset$; $B_1 = B_2 = B$

  for *each* $i \in A_2$ **do**

    Let $j \in \arg\max_{k \in \{1,2\}} v(i|S_k)$

    if $c_i \leq \frac{10B}{x} v(i|S_j) \leq B_j$ **then**

      $S_j = S_j \cup \{i\}$

      $B_j = B_j - \frac{10B}{x} v(i|S_j)$

  for $j \in \{1, 2\}$ **do**

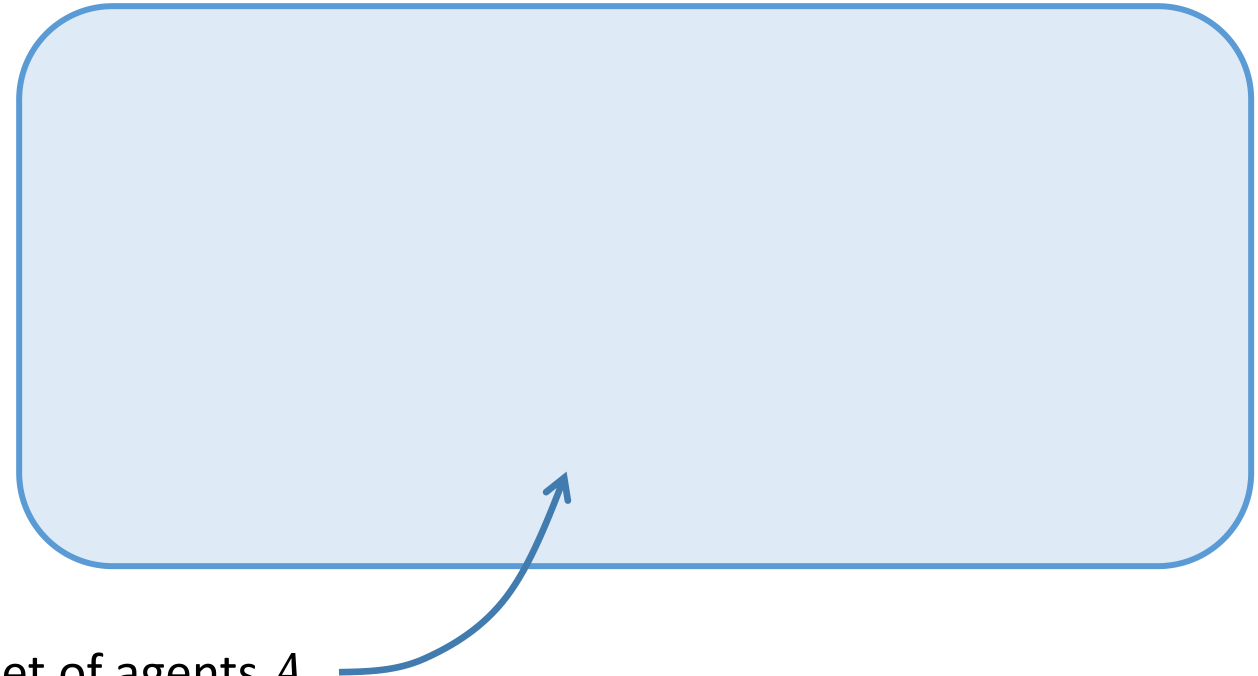
    $T_j = \text{ALG}(S_j)$

  Let S be the best solution among S_1, S_2, T_1, T_2

  return S

Key idea: simultaneous threshold greedy algorithm

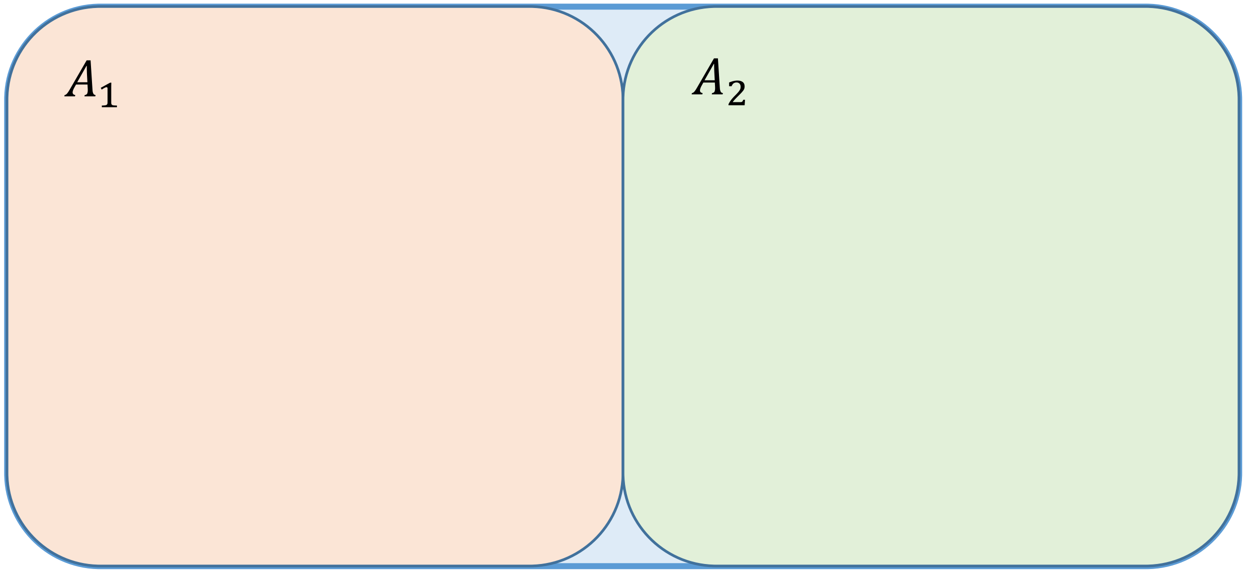
the core algorithmic idea



Initial set of agents A

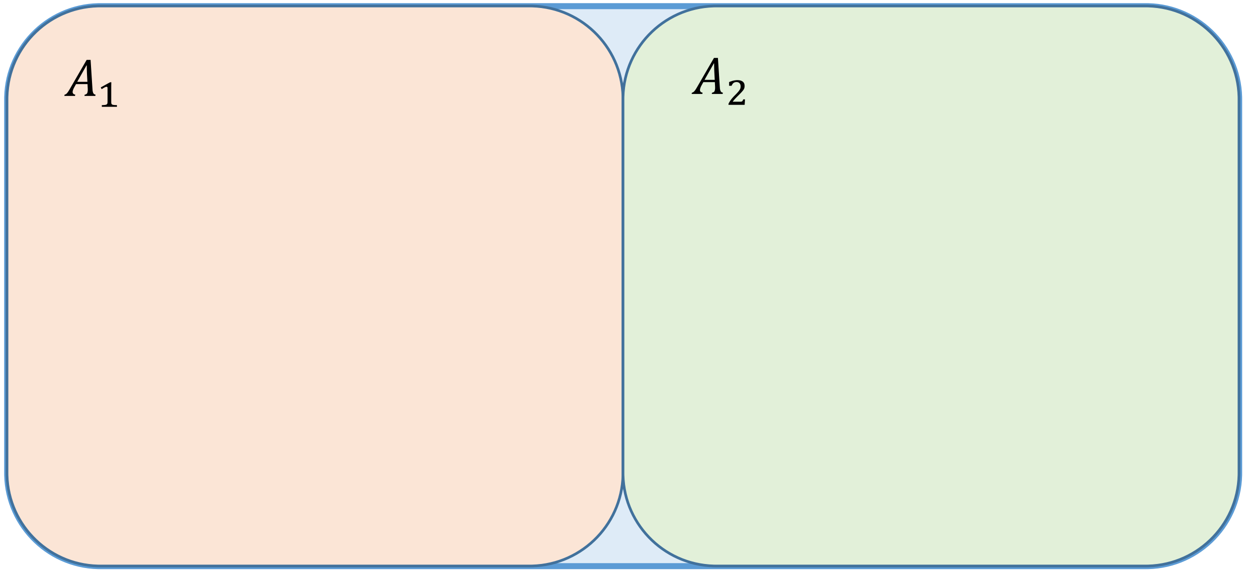
- We randomly split A into A_1 and A_2.

the core algorithmic idea



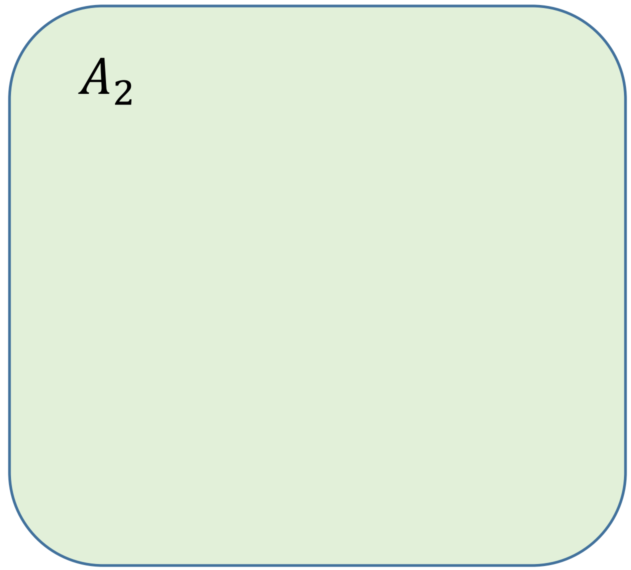
- We randomly split A into A_1 and A_2.

the core algorithmic idea



- We randomly split A into A_1 and A_2.
- We (approximately) solve on A_1 in order to obtain a rough estimate of the optimal solution in A_2.

the core algorithmic idea

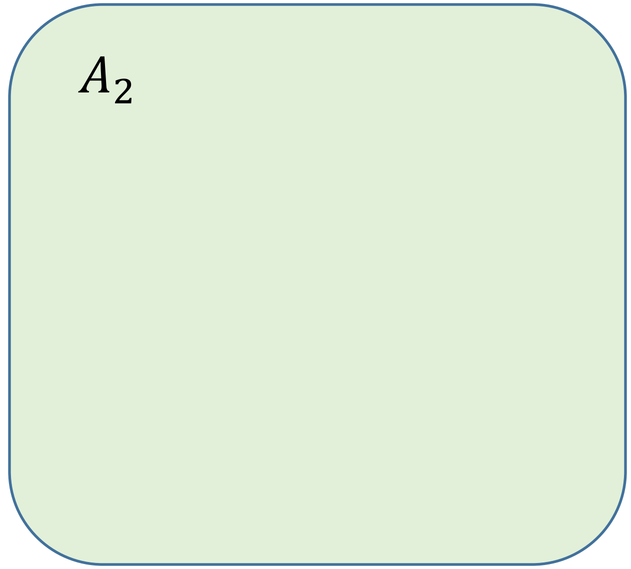


Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea



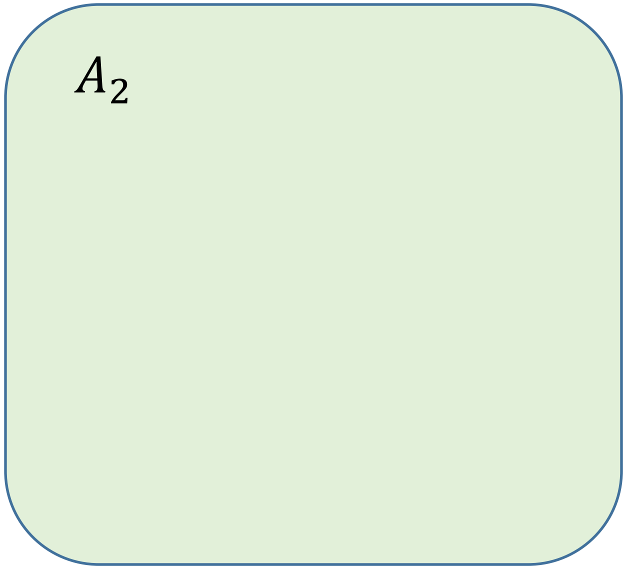
Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

Agent i is efficient.

the core algorithmic idea



Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

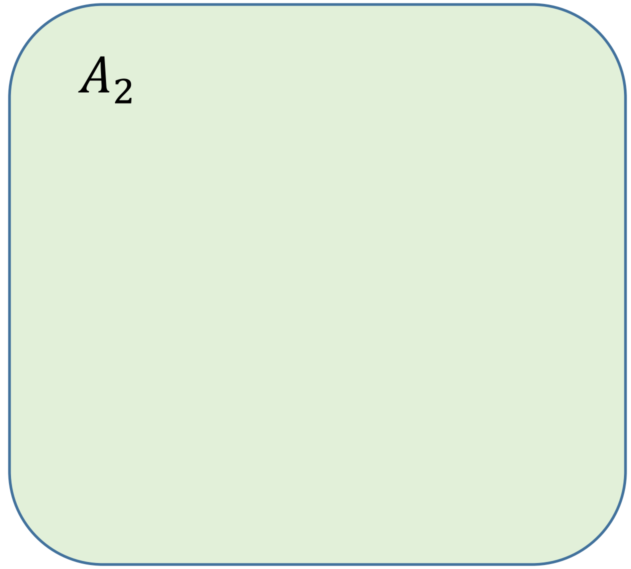
- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.

Enough leftover budget.

- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea

Marginal value
$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$



- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.

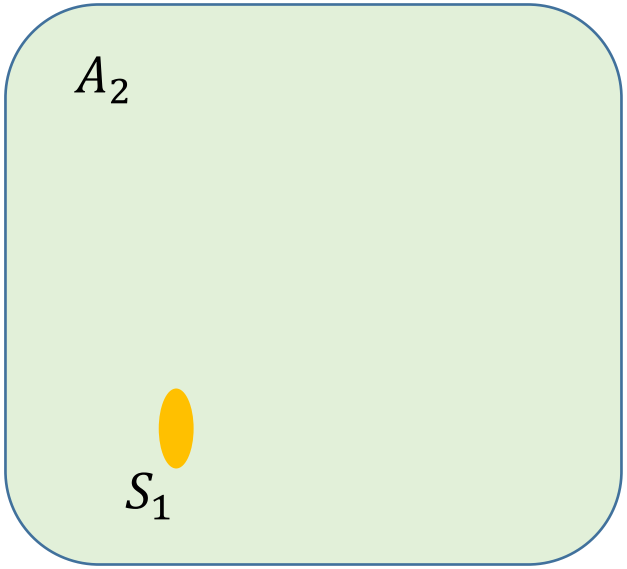
Take it or leave it offer!

- Agent i is added to S_j if $c_i \leq \boxed{10 \frac{v(i|S_j)}{OPT(A_1)} B} \leq B_j$

the core algorithmic idea

Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

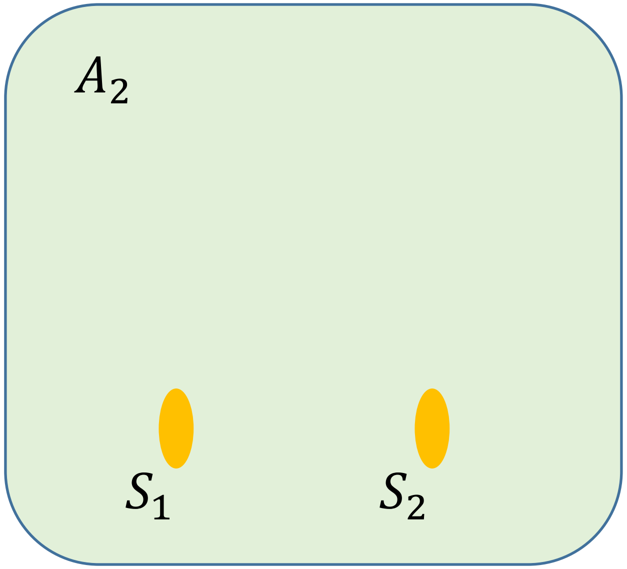


- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea

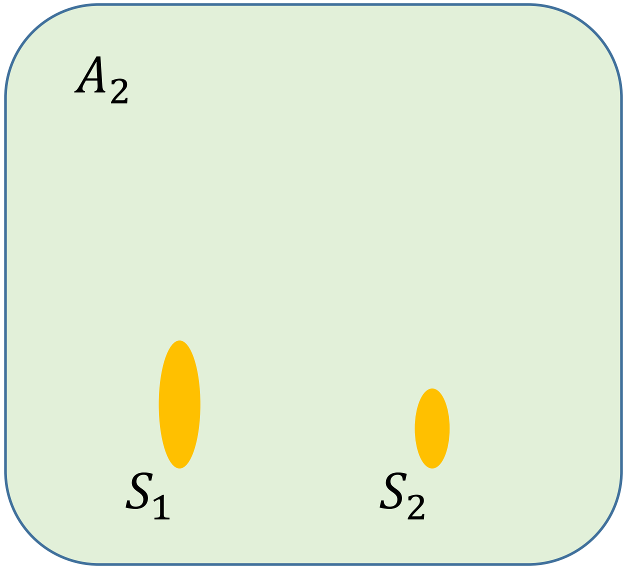
Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$



- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea



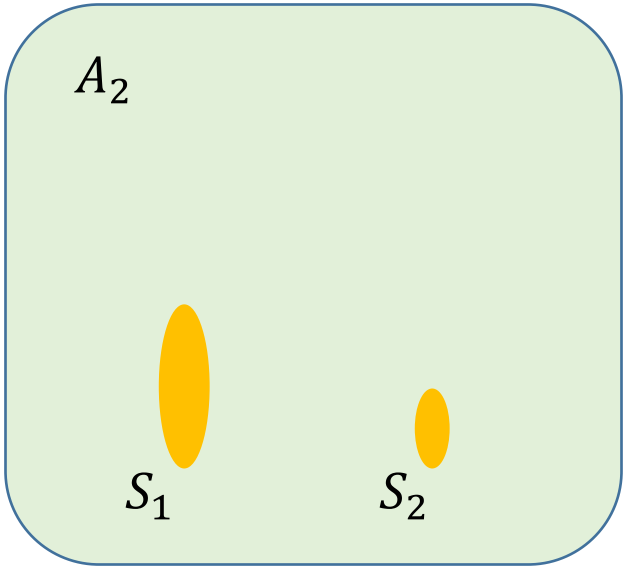
Marginal value
$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea

Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

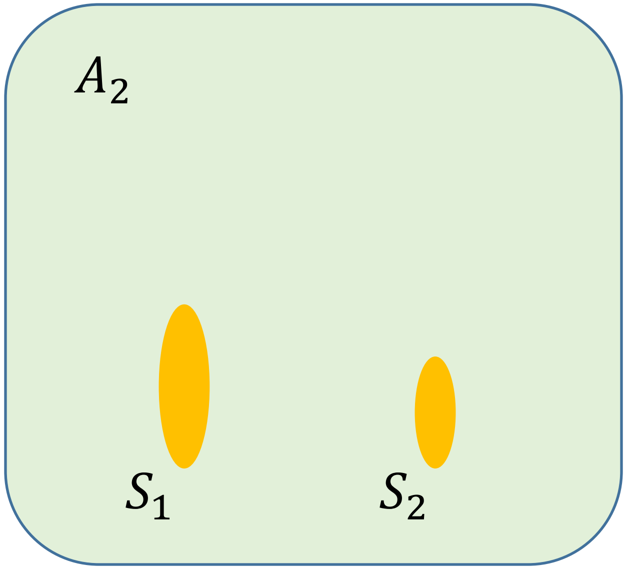


- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea

Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

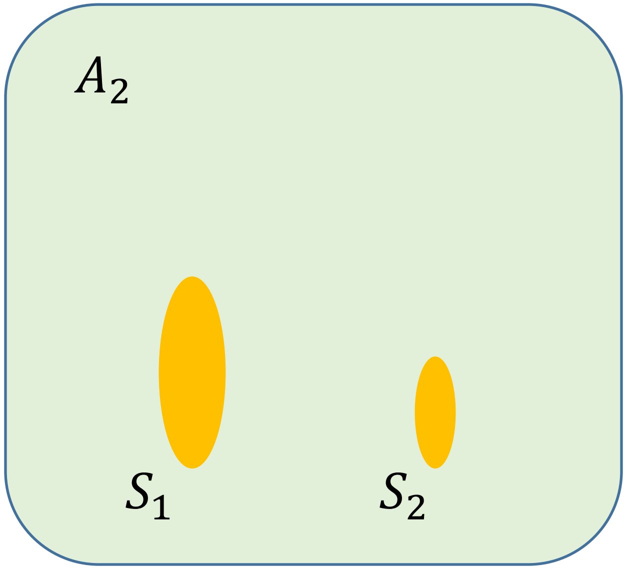


- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea

Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

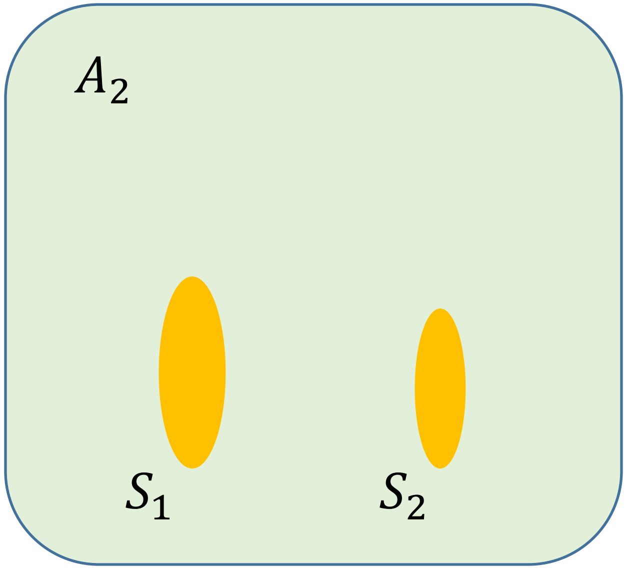


- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea

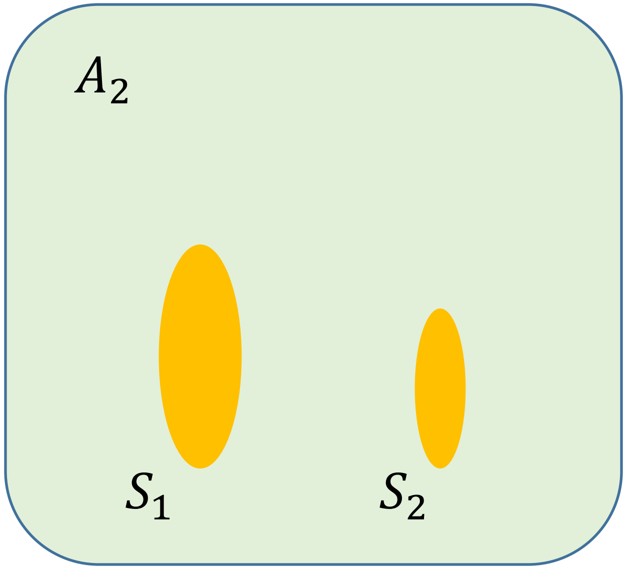
Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$



- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10\,\frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea



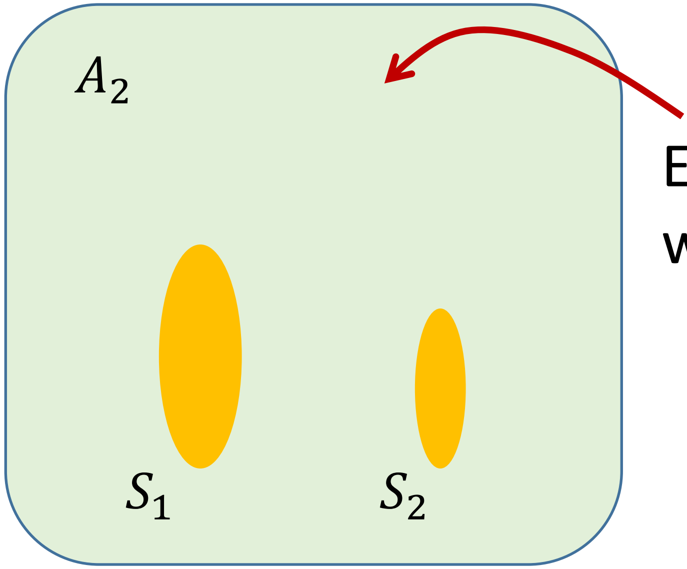
Marginal value

$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$

- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea

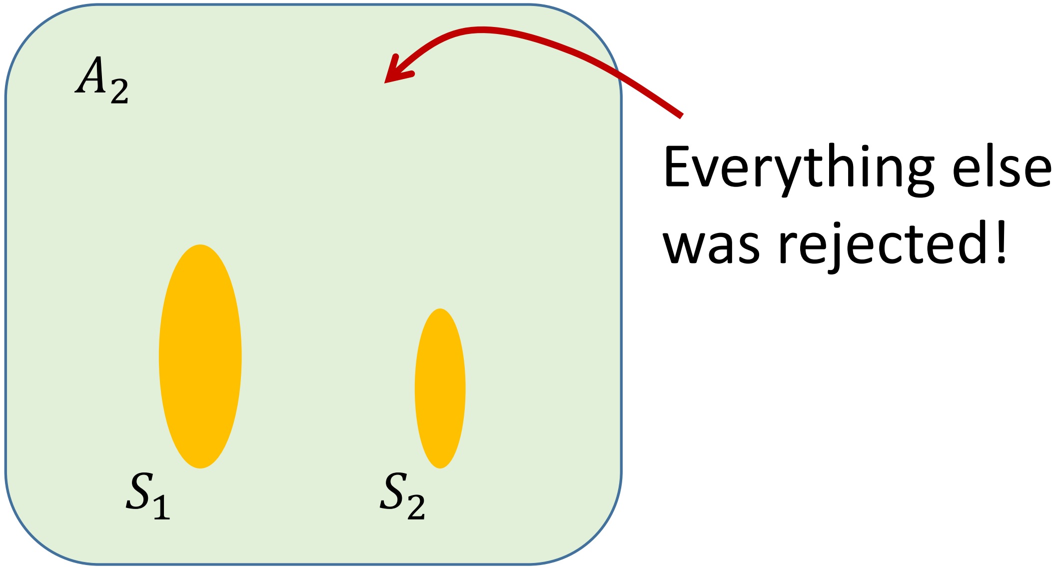
Marginal value
$v(i|S_j) = v(S_j \cup \{i\}) - v(S_j)$



Everything else was rejected!

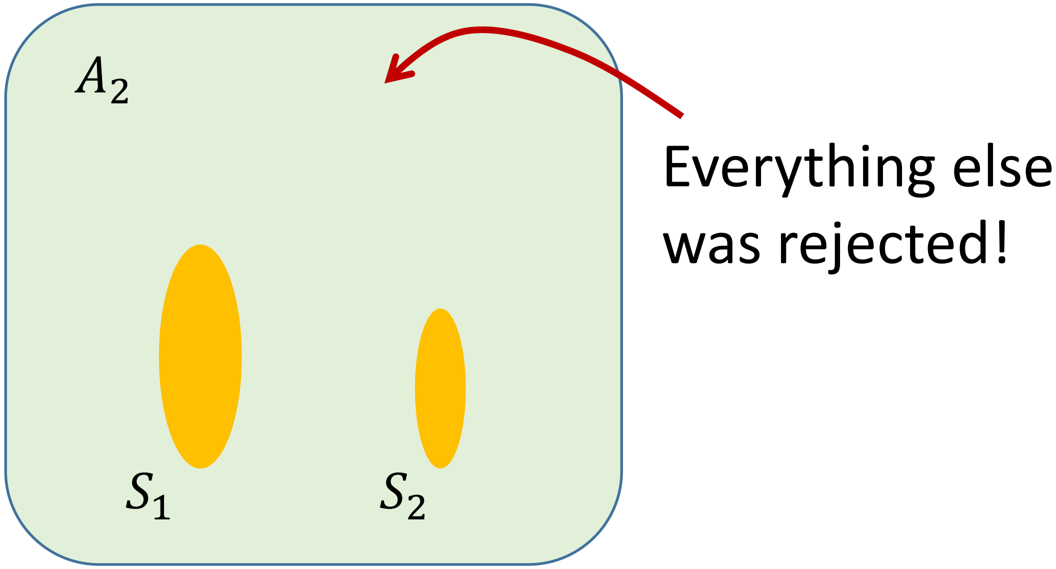
- We build two solutions S_1 and S_2 each with budget B (say B_1, B_2).
- We iterate through the agents once. Each i is a candidate for the solution S_j that maximizes her marginal value.
- Agent i is added to S_j if $c_i \leq 10 \frac{v(i|S_j)}{OPT(A_1)} B \leq B_j$

the core algorithmic idea



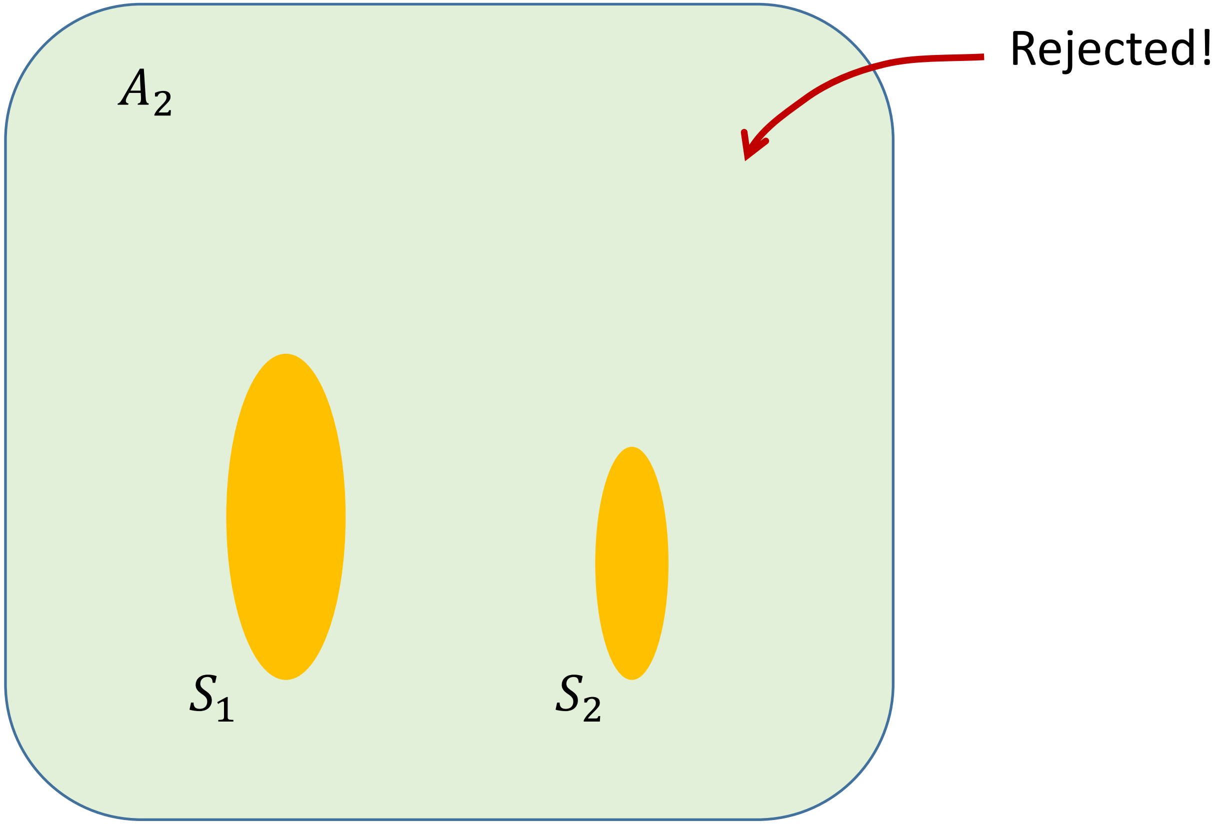
- In the end, we return the best solution contained in of S_1 or S_2

the core algorithmic idea

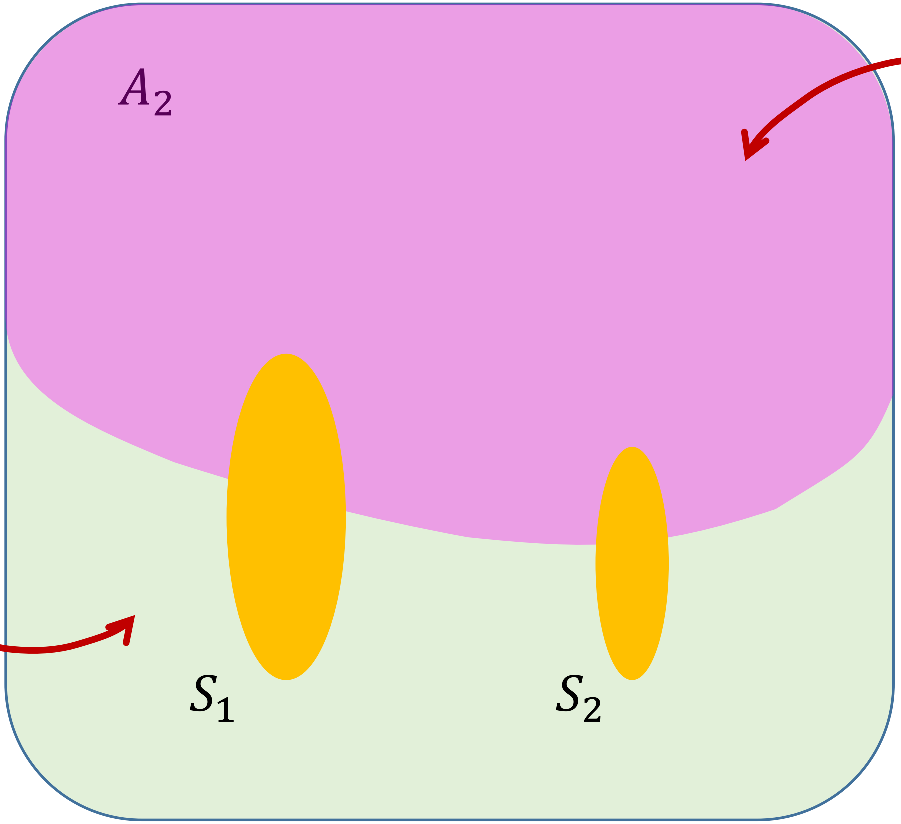


- In the end, we return the best solution contained in of S_1 or S_2
- $p_i = \frac{10B}{OPT(A_1)} \cdot$ (marginal value of i when added)
- The residual budgets B_1, B_2 are defined so that both S_1 and S_2 end up budget-feasible.

approximation ratio



approximation ratio



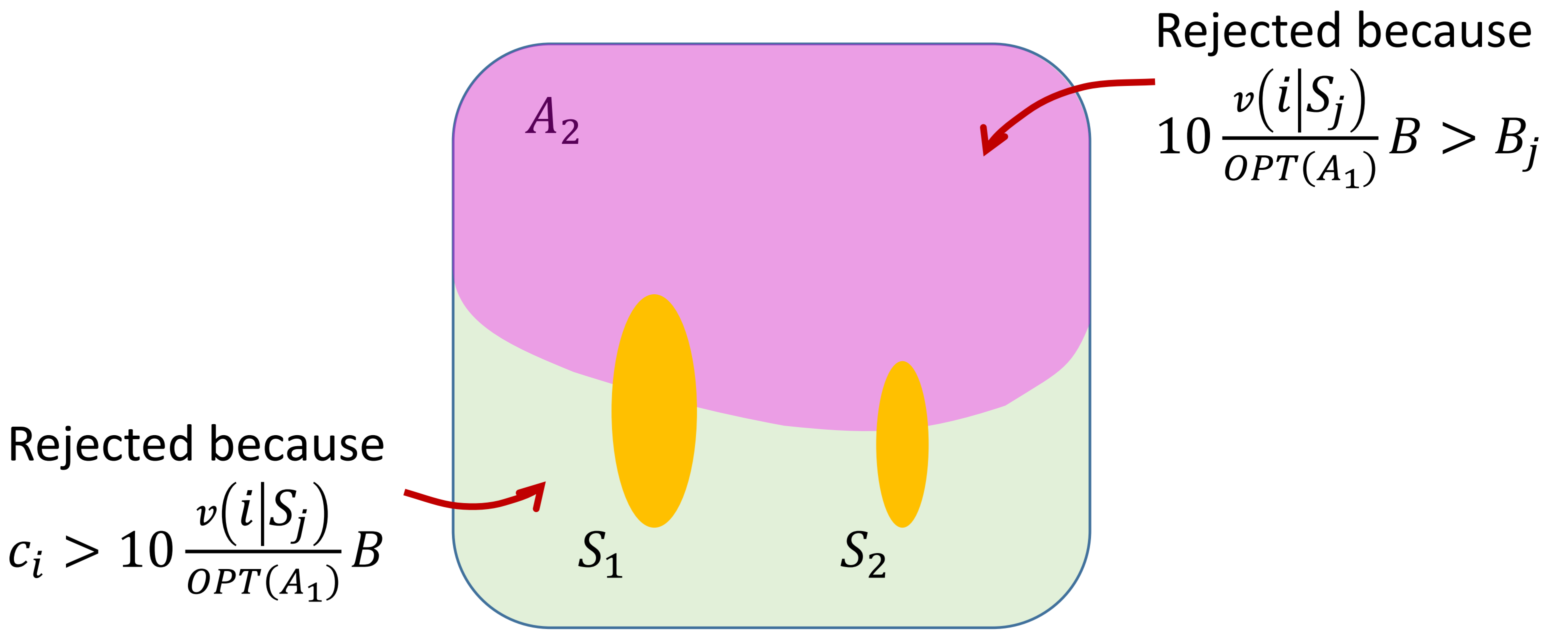
Rejected because

$$10\,\frac{v(i|S_j)}{OPT(A_1)}B > B_j$$

Rejected because

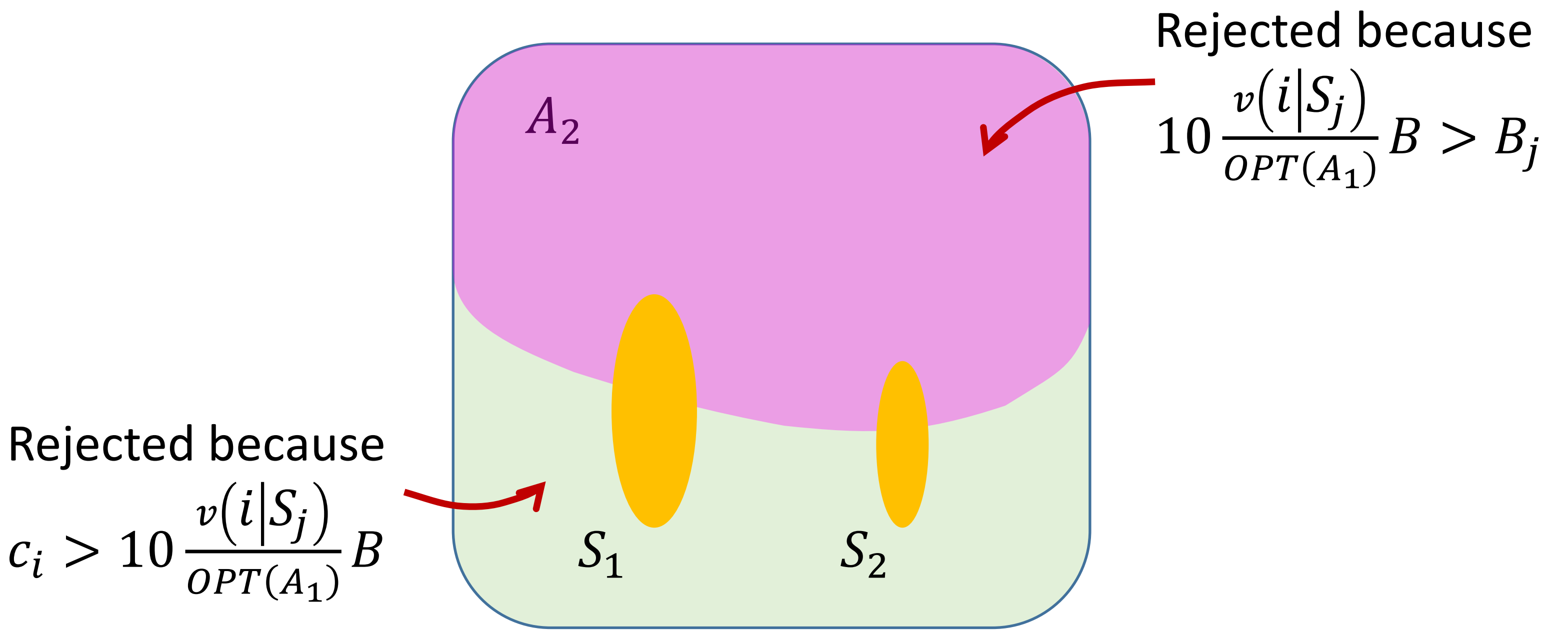
$$c_i > 10\,\frac{v(i|S_j)}{OPT(A_1)}B$$

approximation ratio



- If the purple part is non-empty then at some point we have spent most of the budget of S_1 or S_2.

approximation ratio

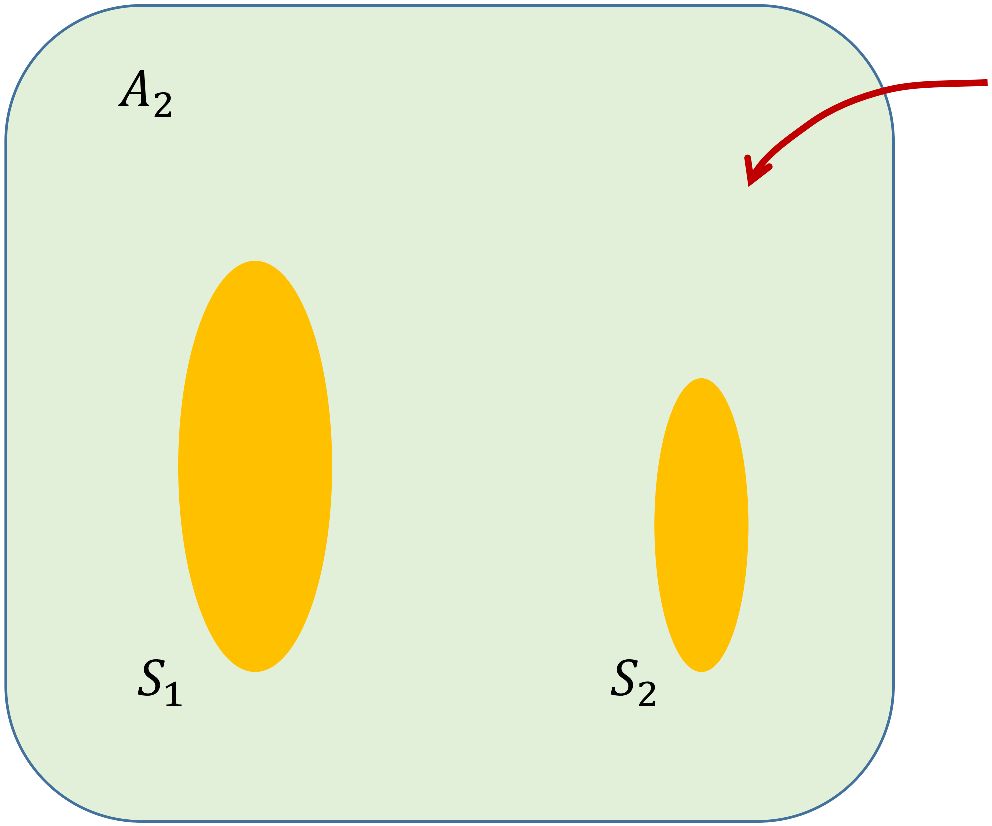


- If the purple part is non-empty then at some point we have spent most of the budget of S_1 or S_2.

With constant probability

- Since we spend at a rate $\approx \frac{10B}{OPT(A_1)} \leq \frac{40B}{OPT(A)}$, this means we bought value $\geq \frac{OPT(A)}{40}$.

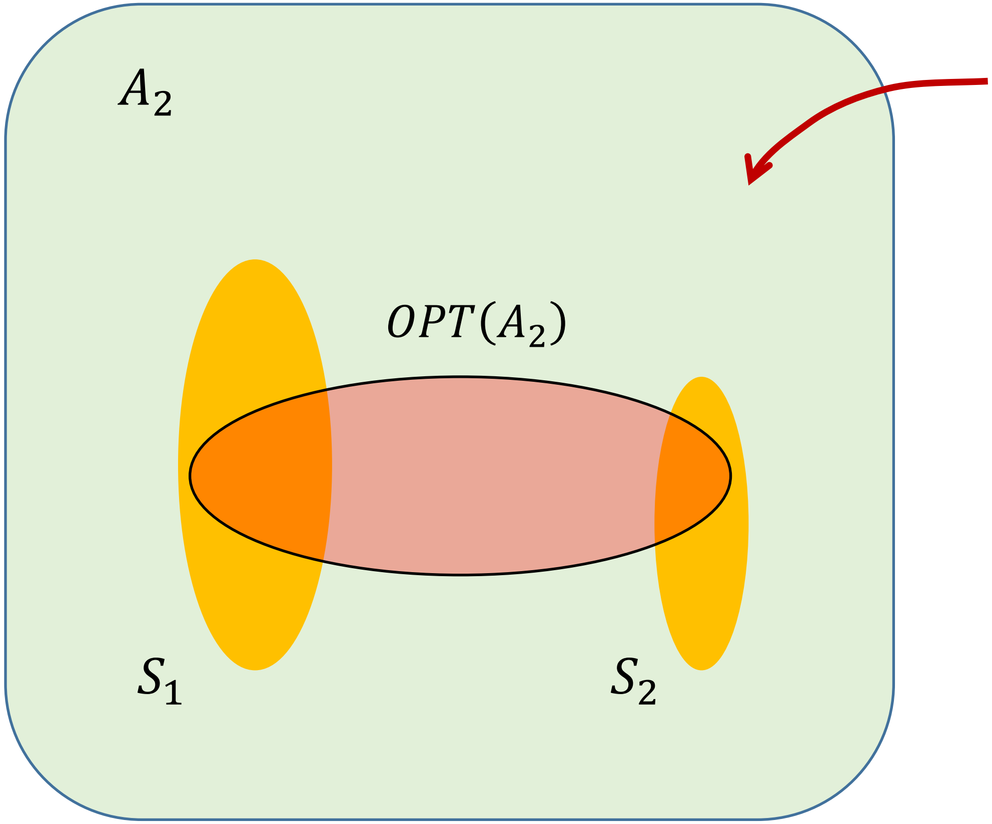
approximation ratio



Suppose everything is rejected because

$$c_i > 10\,\frac{v(i|S_j)}{OPT(A_1)}B$$

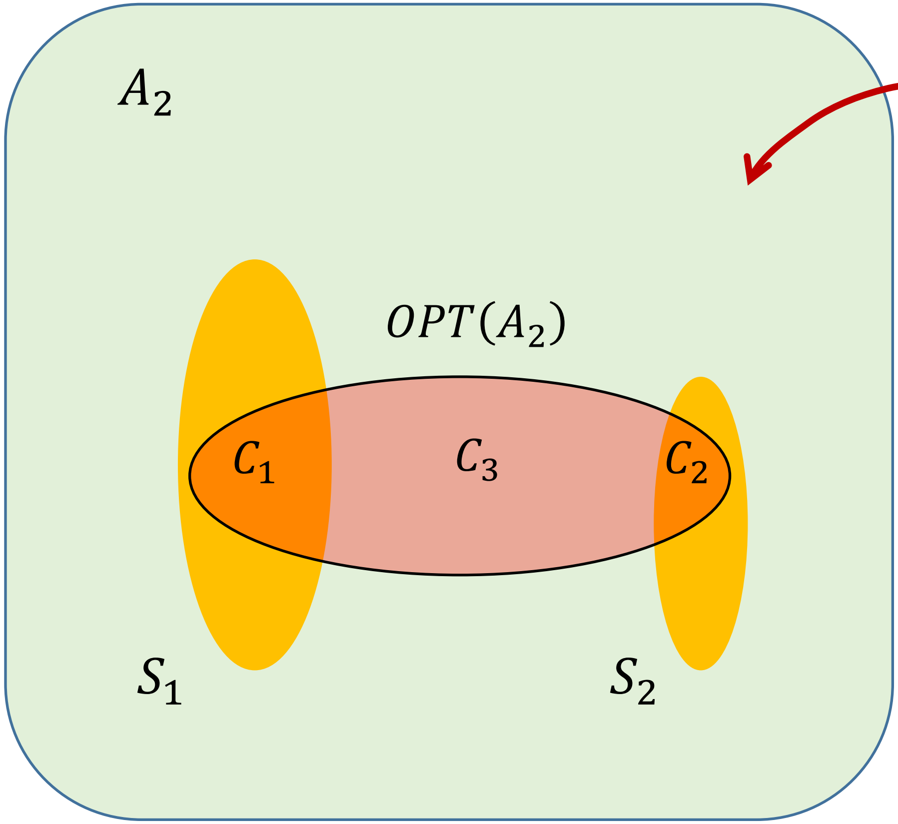
approximation ratio



Suppose everything is rejected because

$$c_i > 10\,\frac{v(i|S_j)}{OPT(A_1)}B$$

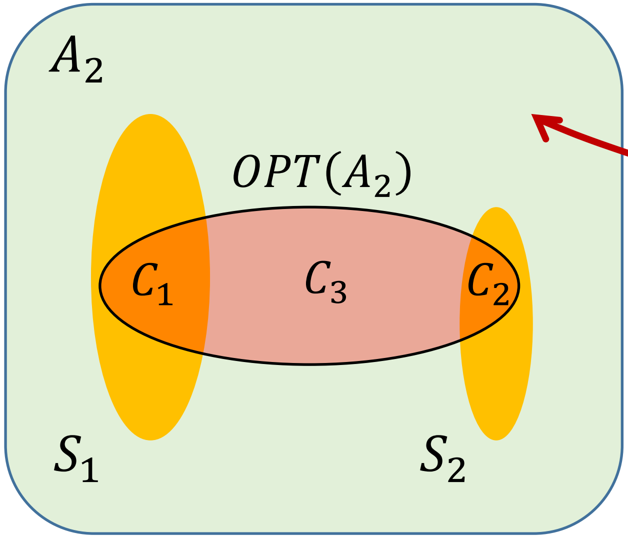
approximation ratio



Suppose everything is rejected because

$$c_i > 10\,\frac{v(i|S_j)}{OPT(A_1)}B$$

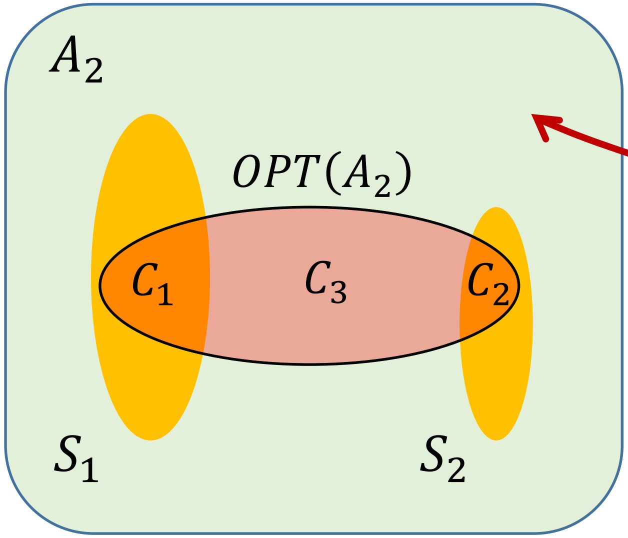
approximation ratio



Everything is rejected because $c_i > 10\,\frac{v(i|S_j)}{OPT(A_1)}B$

- $\frac{OPT(A)}{4} \le OPT(A_2) \le v(C_1) + v(C_2) + v(C_3)$

approximation ratio

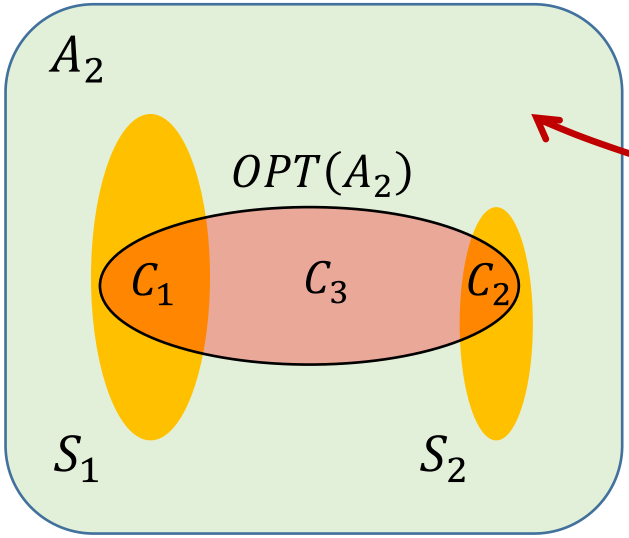


Everything is rejected because
$c_i > 10 \frac{v(i|S_j)}{OPT(A_1)} B$

With constant probability

- $\frac{OPT(A)}{4} \leq OPT(A_2) \leq v(C_1) + v(C_2) + v(C_3)$

approximation ratio



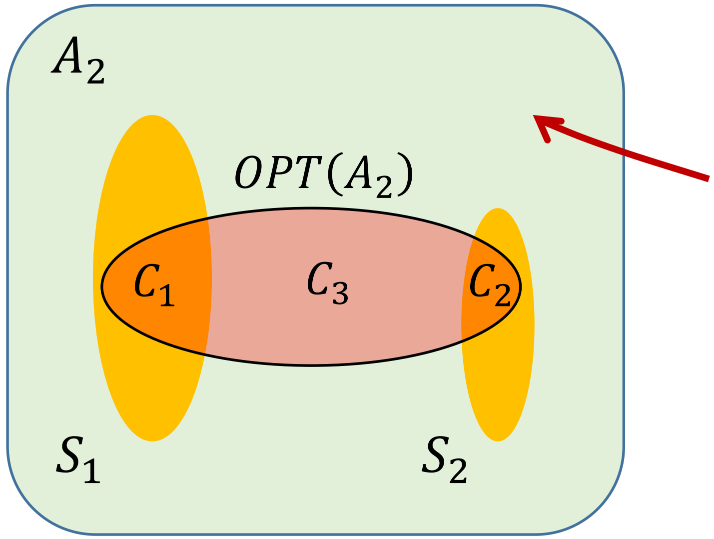
Everything is rejected because $c_i > 10 \frac{v(i|S_j)}{OPT(A_1)} B$

With constant probability

By subadditivity

- $\frac{OPT(A)}{4} \leq OPT(A_2) \leq v(C_1) + v(C_2) + v(C_3)$

approximation ratio

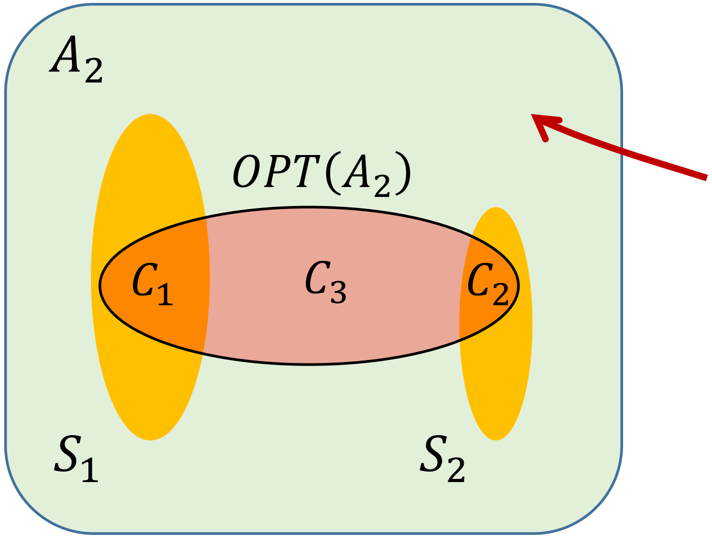


Everything is rejected because

$$c_i > 10\,\frac{v(i|S_j)}{OPT(A_1)}B$$

- $\frac{OPT(A)}{4} \leq OPT(A_2) \leq v(C_1) + v(C_2) + v(C_3)$
- By submodularity: $v(C_3) \leq v(C_3 \cup S_1) + v(C_3 \cup S_2)$

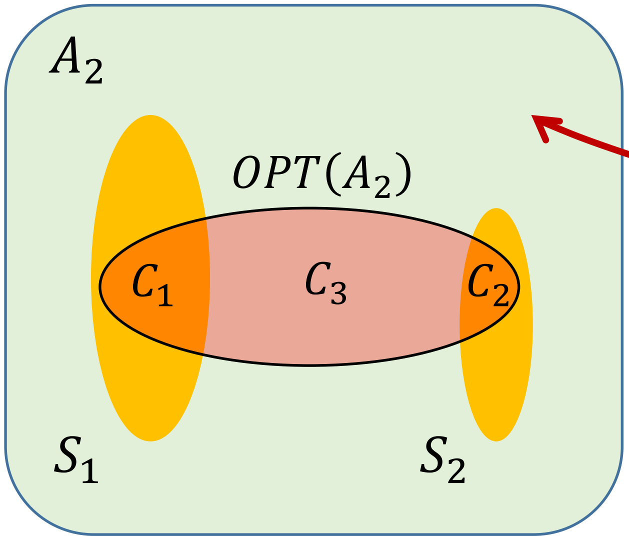
approximation ratio



Everything is rejected because $c_i > 10\,\frac{v(i|S_j)}{OPT(A_1)}B$

- $\frac{OPT(A)}{4} \leq OPT(A_2) \leq v(C_1) + v(C_2) + v(C_3)$
- By submodularity: $v(C_3) \leq v(C_3 \cup S_1) + v(C_3 \cup S_2)$
- Again by submodularity: $v(C_3 \cup S_j) \leq v(S_j) + \frac{OPT(A)}{10}$

approximation ratio



Everything is rejected because
$c_i > 10 \frac{v(i|S_j)}{OPT(A_1)} B$

- Putting them together:

$$\frac{OPT(A)}{20} \leq v(C_1) + v(C_2) + v(S_1) + v(S_2)$$

- So, the (approximately) best solution contained in S_1 or S_2 is a constant fraction of $OPT(A)$.

is this practical?

- A 505-approximation mechanism doesn’t seem like much…

is this practical?

- A 505-approximation mechanism doesn't seem like much…
- Under a large market assumption, the ratio drops to ~ 20.

is this practical?

- A 505-approximation mechanism doesn't seem like much…
- Under a large market assumption, the ratio drops to ~ 20.
- When tested on real and synthetic data, the ratio was < 2.

directions for future work

- Is it possible to design deterministic mechanisms with the same properties?
- Can we achieve approximation guarantees close to those we know for the algorithmic counterparts of these problems?
- Better for restricted families of objectives, e.g., cut functions on directed graphs?
- Are there stronger negative results? Separation of randomized and deterministic mechanisms w.r.t. the number of queries?

directions for future work

- Is it possible to design deterministic mechanisms with the same properties?
- Can we achieve approximation guarantees close to those we know for the algorithmic counterparts of these problems?
- Better for restricted families of objectives, e.g., cut functions on directed graphs?
- Are there stronger negative results? Separation of randomized and deterministic mechanisms w.r.t. the number of queries?

thank you!