Cost Sharing and Approximation Algorithms

— Lecture 2 —

Guido Schäfer
CWI Amsterdam / VU University Amsterdam
g.schaefer@cwi.nl

ADFOCS 2010
11th Max Planck Advanced Course on the Foundations of Computer Science
August 2–6, 2010, Saarbrücken, Germany

Recap

Moulin Mechanisms:

- realize strong notion of group-strategyproofness
- driven by cross-monotonic cost sharing schemes
- example: Steiner forest (by-products: new insights, algorithm, LP formulation)

Trade-Off Group-Strategyproofness vs. Approximation:

- constant budget balance and polylogarithmic social cost factors for Steiner tree, Steiner forest, facility location
- gap between best achievable approximation guarantee and budget balance factor of Moulin mechanisms (sometimes significant!)

Moulin Mechanisms: Limitations and New Trade-Offs

Inefficiency of Moulin Mechanisms

Moulin mechanisms may have poor budget balance or social cost approximation guarantees

Examples:

	β	α
vertex cover	n^c	$\Omega(\log n)$
set cover	n	$\Omega(\log n)$
facility location	3	$\Omega(\log n)$
Steiner tree	2	$\Omega(\log^2 n)$
makespan scheduling	2	$\Omega(\log n)$

Limitations of Moulin Mechanisms

Theorem

Suppose there is a set $S \subseteq U$ such that

$$C(S) \geq \beta \cdot \sum_{i \in S} C(\{i\}).$$

Then there is no Moulin mechanism that is $(\beta - \varepsilon)$-budget balance for any $\varepsilon > 0$.

[Brenner, Schäfer, TCS '08]

Example: Completion Time Scheduling

Minimum Completion Time Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- completion time of job i: C_i
- **Goal:** compute schedule such that $\sum_i C_i$ is minimized

Consequence: $(n+1)/2$ lower bound on budget balance for minimum completion time scheduling problem $1|p_i = 1|\sum_i C_i$

Example: Completion Time Scheduling

Minimum Completion Time Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- completion time of job i: C_i
- **Goal:** compute schedule such that $\sum_i C_i$ is minimized

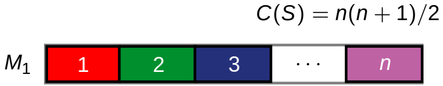
Consequence: $(n+1)/2$ lower bound on budget balance for minimum completion time scheduling problem $1|p_i = 1|\sum_i C_i$

Example: Completion Time Scheduling

Minimum Completion Time Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- completion time of job i: C_i
- **Goal:** compute schedule such that $\sum_i C_i$ is minimized

Consequence: $(n+1)/2$ lower bound on budget balance for minimum completion time scheduling problem $1|p_i = 1|\sum_i C_i$



Example: Completion Time Scheduling

Minimum Completion Time Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- completion time of job i: C_i
- **Goal:** compute schedule such that $\sum_i C_i$ is minimized

Consequence: $(n+1)/2$ lower bound on budget balance for minimum completion time scheduling problem $1|p_i = 1|\sum_i C_i$

Limitations of Moulin Mechanisms

Theorem

Suppose that

$$C(S) \geq \frac{1}{\delta} \cdot C(U) \quad \forall S \subseteq U,\ S \neq \emptyset.$$

Then there exists no Moulin mechanism that is $(\frac{H_n}{\delta} - \varepsilon)$-approximate for any $\varepsilon > 0$.

[Brenner, Schäfer, TCS '08]

Example: Makespan Scheduling

Minimum Makespan Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- makespan: maximum completion time over all jobs
- **Goal:** compute schedule that minimizes makespan

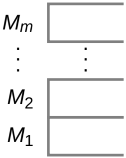
Consequence: H_n lower bound on social cost approximation for minimum makespan problem $P|p_i = 1|C_{\max}$

Example: Makespan Scheduling

Minimum Makespan Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- makespan: maximum completion time over all jobs
- **Goal:** compute schedule that minimizes makespan

Consequence: H_n lower bound on social cost approximation for minimum makespan problem $P|p_i = 1|C_{\max}$

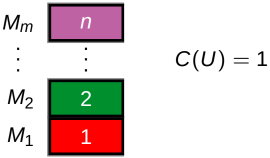


Example: Makespan Scheduling

Minimum Makespan Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- makespan: maximum completion time over all jobs
- **Goal:** compute schedule that minimizes makespan

Consequence: H_n lower bound on social cost approximation for minimum makespan problem $P|p_i = 1|C_{\max}$

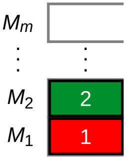


Example: Makespan Scheduling

Minimum Makespan Scheduling Problem:

- set of n jobs, job i has processing time p_i
- m identical machines, no preemption
- makespan: maximum completion time over all jobs
- **Goal:** compute schedule that minimizes makespan

Consequence: H_n lower bound on social cost approximation for minimum makespan problem $P|p_i = 1|C_{\max}$



$C(\{1, \ldots, i\}) = 1$

Public Excludable Good

Public Excludable Good Problem:

$$C(S) = 1 \quad \forall S \subseteq U,\ S \neq \emptyset \quad \text{and} \quad C(\emptyset) = 0$$

Examples:

- minimum spanning tree, Steiner tree, Steiner forest
- vertex cover, set cover, facility location
- makespan scheduling

Theorem

Every truthful mechanism for the public excludable good problem that is β-budget balanced is no better than $\Omega(\log n/\beta)$-approximate.

[Dobzinski, Mehta, Roughgarden, Sundararajan, SAGT '08]

New Trade-Offs

Group-Strategyproofness:

- very strong notion of truthfulness
- often bottleneck in achieving good approximation guarantees
- strong lower bounds exist (even if we allow exponential time)

Idea: consider weaker notions of group-strategyproofness, without sacrificing coalitional game theory viewpoint
$\Rightarrow$ weak group-strategyproofness

[Mehta, Roughgarden, Sundararajan, GEB '09]

Illustration: Weak Group-Strategyproofness

Definition

A cost sharing mechanism M is weakly group-strategyproof iff for all $S \subseteq U$

$$\exists i \in S: \; u_i(\tilde{q},\tilde{p}) \leq u_i(q,p)$$

(q,p): outcome if $b_i = v_i$ for every $i \in S$
$(\tilde{q},\tilde{p})$: outcome if $b_i = \;\cdot\;$ for every $i \in S$

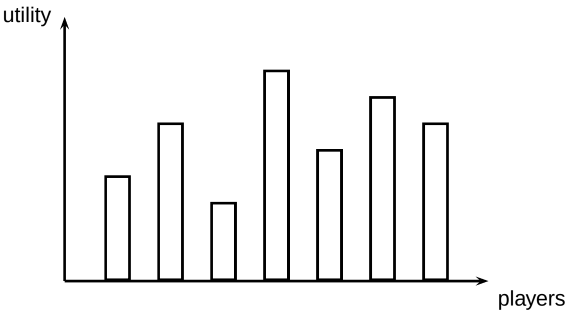


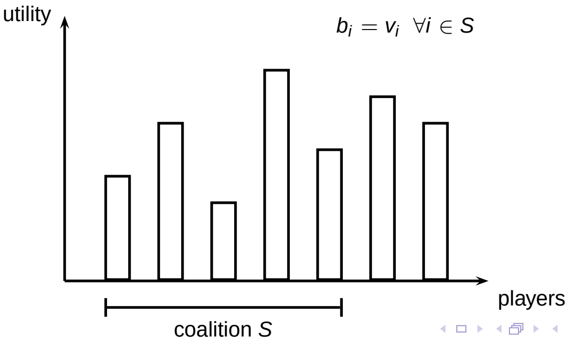
Illustration: Weak Group-Strategyproofness

Definition

A cost sharing mechanism M is weakly group-strategyproof iff for all $S \subseteq U$

$$\exists i \in S : \; u_i(\tilde{q}, \tilde{p}) \leq u_i(q, p)$$

(q, p): outcome if $b_i = v_i$ for every $i \in S$
$(\tilde{q}, \tilde{p})$: outcome if $b_i = \cdot$ for every $i \in S$

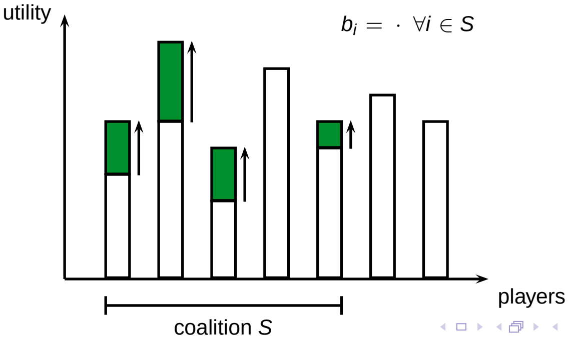


Definition

A cost sharing mechanism M is weakly group-strategyproof iff for all $S \subseteq U$

$$\exists i \in S: \; u_i(\tilde{q}, \tilde{p}) \leq u_i(q, p)$$

(q,p): outcome if $b_i = v_i$ for every $i \in S$
$(\tilde{q},\tilde{p})$: outcome if $b_i = \cdot$ for every $i \in S$



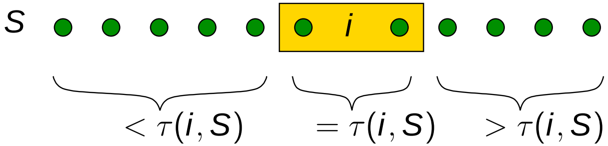
Acyclic Mechanisms

Valid Offer Function

Offer Function: $\tau : U \times 2^U \to \mathbb{R}^+$
$\tau(i, S) =$ offer time of player i with respect to $S \subseteq U$

Valid Offer Function: τ is valid for a cost sharing function ξ if for every subset $S \subseteq U$ and every player $i \in S$:

1. $\xi_i(S \setminus T) = \xi_i(S) \quad \forall T \subseteq G(i, S)$
2. $\xi_i(S \setminus T) \geq \xi_i(S) \quad \forall T \subseteq G(i, S) \cup (E(i, S) \setminus \{i\})$

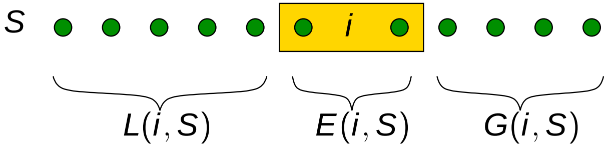


Valid Offer Function

Offer Function: $\tau : U \times 2^U \to \mathbb{R}^+$
$\tau(i, S) =$ offer time of player i with respect to $S \subseteq U$

Valid Offer Function: τ is valid for a cost sharing function ξ if for every subset $S \subseteq U$ and every player $i \in S$:

1. $\xi_i(S \setminus T) = \xi_i(S) \quad \forall T \subseteq G(i, S)$
2. $\xi_i(S \setminus T) \geq \xi_i(S) \quad \forall T \subseteq G(i, S) \cup (E(i, S) \setminus \{i\})$



Acyclic Mechanism $M(\xi, \tau)$:

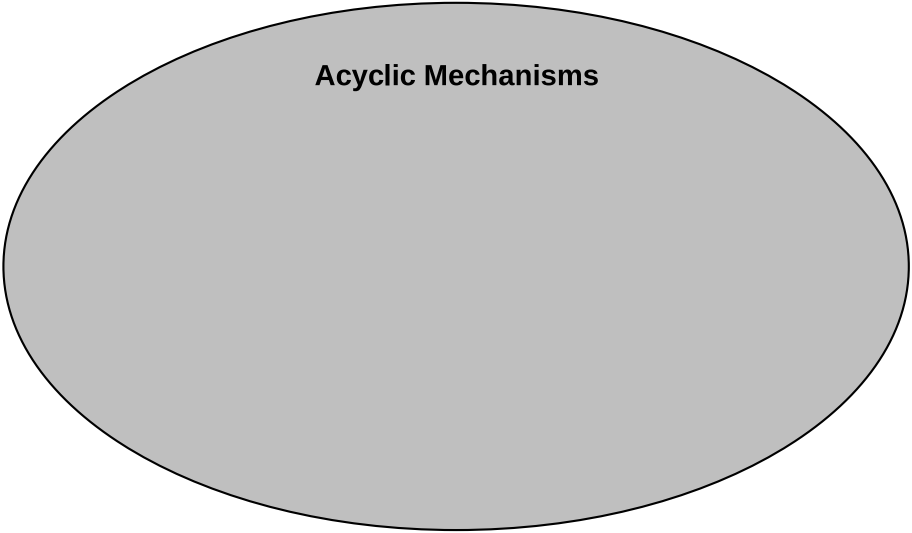
1: Initialize: $Q \leftarrow U$
2: If for each player $i \in Q$: $\xi_i(Q) \leq b_i$ then STOP
3: Otherwise: Among all players in Q with $\xi_i(Q) > b_i$, let i^* be one with minimum offer time $\tau(i, Q)$. Remove i^* from Q and repeat.

Theorem

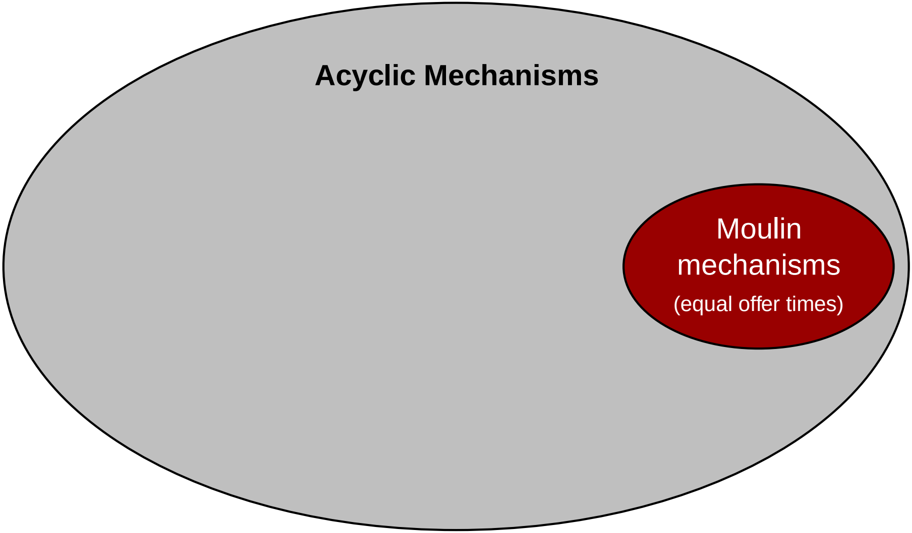
If τ is a valid offer function for ξ, then the acyclic mechanism $M(\xi, \tau)$ is weakly group-strategyproof.

[Mehta, Roughgarden, Sundararajan, GEB '09]

Universe of Acyclic Mechanisms



Universe of Acyclic Mechanisms



Known Results

Several primal-dual algorithms naturally give rise to valid offer functions.

Acyclic Mechanisms:

	β	α	Moulin (β)
vertex cover	2	$O(\log n)$	n^c
set cover	$O(\log n)$	$O(\log n)$	n
facility location	1.61	$O(\log n)$	3
Steiner tree	2	$O(\log^2 n)$	2

[Mehta, Roughgarden, Sundararajan, GEB '09]

Generalized Incremental Mechanisms

Design of Cost Sharing Mechanisms

Most Previous Cost Sharing Mechanisms:

- developed in case-by-case studies
- driven by cost sharing schemes that need to satisfy certain properties (cross-monotonicity, valid offer function)
  $\Rightarrow$ problem-specific and often non-trivial task

Question: Can we devise a framework that allows to derive truthful cost sharing mechanisms from existing approximation algorithms?

Framework

Let *ALG* be a ρ-approximation algorithm for the optimization problem $\mathcal{P}$.

Theorem

There is a weakly group-strategegproof and ρ-budget balanced cost sharing mechanism.

[Brenner, Schäfer, SAGT '08]

Advantages:

- weakly group-strategyproofness comes for free
- mechanism inherits approximation guarantee
- approximation algorithm is used as a black-box

Disadvantage: mechanism does not necessarily satisfy the no positive transfer property

Framework

Order Function: $\tau : U \times 2^U \to \mathbb{R}^+$

$\tau(i, S) =$ unique offer time of player i with respect to $S \subseteq U$

Generalized Incremental Mechanism $M(ALG, \tau)$**:**

```
Initialize: A ← ∅, R ← U
while A ≠ R do
    Let i be the player with minimum τ(i, R) among R \ A
    Define ξ_i := C̄(A ∪ {i}) − C̄(A)    (marginal cost)
    if ξ_i ≤ b_i then A ← A ∪ {i} else R ← R \ {i}
end
Output the characteristic vector of A and payments ξ
```

Note: no positive transfer property holds if approximate cost is monotone increasing, i.e., $\bar{C}(S) \leq \bar{C}(T)$ for all $S \subseteq T \subseteq U$

Budget Balance and WGSP

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is ρ-budget balanced and weakly group-strategyproof.

Proof:

In every iteration, we have $\sum_{i \in A} \xi_i = \bar{C}(A)$. ρ-budget balance follows from the approximation guarantee of ALG.

Fix a coalition $S \subseteq U$ and consider the runs of $M(ALG, \tau)$ on (b_{-S}, b'_S) and (b_{-S}, v_S). These runs are identical until first player in S, say i, is considered. The payment ξ_i of i only depends on the set of previously accepted players, which is the same in both runs. Player i cannot gain by reporting b'_i instead of v_i. □

Budget Balance and WGSP

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is ρ-budget balanced and weakly group-strategyproof.

Proof:

In every iteration, we have $\sum_{i \in A} \xi_i = \bar{C}(A)$. ρ-budget balance follows from the approximation guarantee of ALG.

Fix a coalition $S \subseteq U$ and consider the runs of $M(ALG, \tau)$ on (b_{-S}, b'_S) and (b_{-S}, v_S). These runs are identical until first player in S, say i, is considered. The payment ξ_i of i only depends on the set of previously accepted players, which is the same in both runs. Player i cannot gain by reporting b'_i instead of v_i. □

Budget Balance and WGSP

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is ρ-budget balanced and weakly group-strategyproof.

Proof:

In every iteration, we have $\sum_{i \in A} \xi_i = \bar{C}(A)$. ρ-budget balance follows from the approximation guarantee of ALG.

Fix a coalition $S \subseteq U$ and consider the runs of $M(ALG, \tau)$ on (b_{-S}, b'_S) and (b_{-S}, v_S). These runs are identical until first player in S, say i, is considered. The payment ξ_i of i only depends on the set of previously accepted players, which is the same in both runs. Player i cannot gain by reporting b'_i instead of v_i. □

Budget Balance and WGSP

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is ρ-budget balanced and weakly group-strategyproof.

Proof:
In every iteration, we have $\sum_{i \in A} \xi_i = \bar{C}(A)$. ρ-budget balance follows from the approximation guarantee of ALG.

Fix a coalition $S \subseteq U$ and consider the runs of $M(ALG, \tau)$ on (b_{-S}, b'_S) and (b_{-S}, v_S). These runs are identical until first player in S, say i, is considered. The payment ξ_i of i only depends on the set of previously accepted players, which is the same in both runs. Player i cannot gain by reporting b'_i instead of v_i. □

Budget Balance and WGSP

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is ρ-budget balanced and weakly group-strategyproof.

Proof:

In every iteration, we have $\sum_{i \in A} \xi_i = \bar{C}(A)$. ρ-budget balance follows from the approximation guarantee of ALG.

Fix a coalition $S \subseteq U$ and consider the runs of $M(ALG, \tau)$ on (b_{-S}, b'_S) and (b_{-S}, v_S). These runs are identical until first player in S, say i, is considered. The payment ξ_i of i only depends on the set of previously accepted players, which is the same in both runs. Player i cannot gain by reporting b'_i instead of v_i. □

Budget Balance and WGSP

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is ρ-budget balanced and weakly group-strategyproof.

Proof:

In every iteration, we have $\sum_{i \in A} \xi_i = \bar{C}(A)$. ρ-budget balance follows from the approximation guarantee of ALG.

Fix a coalition $S \subseteq U$ and consider the runs of $M(ALG, \tau)$ on (b_{-S}, b'_S) and (b_{-S}, v_S). These runs are identical until first player in S, say i, is considered. The payment ξ_i of i only depends on the set of previously accepted players, which is the same in both runs. Player i cannot gain by reporting b'_i instead of v_i. $\square$

Budget Balance and WGSP

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is ρ-budget balanced and weakly group-strategyproof.

Proof:

In every iteration, we have $\sum_{i \in A} \xi_i = \bar{C}(A)$. ρ-budget balance follows from the approximation guarantee of ALG.

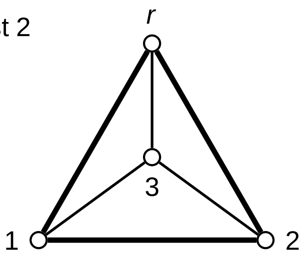
Fix a coalition $S \subseteq U$ and consider the runs of $M(ALG, \tau)$ on (b_{-S}, b'_S) and (b_{-S}, v_S). These runs are identical until first player in S, say i, is considered. The payment ξ_i of i only depends on the set of previously accepted players, which is the same in both runs. Player i cannot gain by reporting b'_i instead of v_i. $\square$

Monotone Approximate Cost

Problem: approximate cost is often not monotone!

Example: Minimum Spanning Tree Game

bold edges have cost 2
all others $1 + \varepsilon$



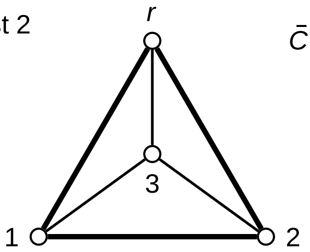
But: marginal approximate cost is increasing if we add players according to Prim's order!

Monotone Approximate Cost

Problem: approximate cost is often not monotone!

Example: Minimum Spanning Tree Game

bold edges have cost 2
all others $1 + \varepsilon$



$\bar{C}(\{1, 2, 3\}) = 3 + 3\varepsilon$

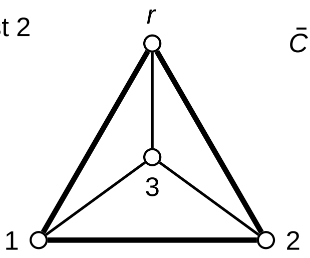
But: marginal approximate cost is increasing if we add players according to Prim's order!

Monotone Approximate Cost

Problem: approximate cost is often not monotone!

Example: Minimum Spanning Tree Game

bold edges have cost 2
all others $1 + \varepsilon$



$$\bar{C}(\{1, 2, 3\}) = 3 + 3\varepsilon$$
$$\bar{C}(\{1, 2\}) = 4$$

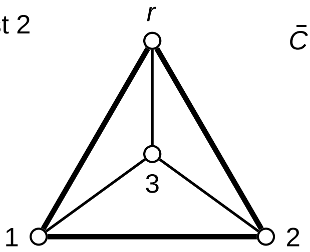
But: marginal approximate cost is increasing if we add players according to Prim's order!

Monotone Approximate Cost

Problem: approximate cost is often not monotone!

Example: Minimum Spanning Tree Game

bold edges have cost 2
all others $1 + \varepsilon$



$\bar{C}(\{1, 2, 3\}) = 3 + 3\varepsilon$

$\bar{C}(\{1, 2\}) = 4$

But: marginal approximate cost is increasing if we add players according to Prim's order!

Two Crucial Ingredients

Consistent Order Function: for every $S \subseteq T$:

T	1	2	3	4	5	6	7	8	9	($\tau(\cdot, T)$ order)
S	1	2	3		5	6		8	9	($\tau(\cdot, T)$ order)

τ-Increasing: ALG is τ-increasing if for every $S \subseteq U$ and every $1 \leq i \leq |S|$:

$$\bar{C}(S_i) - \bar{C}(S_{i-1}) \geq 0,$$

where S_i is the set of the first i elements of S (ordered according to $\tau(\cdot, S)$).

Two Crucial Ingredients

Consistent Order Function: for every $S \subseteq T$:

T	1	2	3	4	5	6	7	8	9	($\tau(\cdot, T)$ order)
S	1	2	3	●	5	6		8	9	($\tau(\cdot, T)$ order)

τ-Increasing: ALG is τ-increasing if for every $S \subseteq U$ and every $1 \leq i \leq |S|$:

$$\bar{C}(S_i) - \bar{C}(S_{i-1}) \geq 0,$$

where S_i is the set of the first i elements of S (ordered according to $\tau(\cdot, S)$).

Two Crucial Ingredients

Consistent Order Function: for every $S \subseteq T$:

T	**1**	**2**	**3**	4	5	6	7	8	9	($\tau(\cdot, T)$ order)
S	1	2	3	●	5	6		8	9	($\tau(\cdot, T)$ order)
S	**1**	**2**	**3**	9	8	5	6			($\tau(\cdot, S)$ order)

τ-Increasing: ALG is τ-increasing if for every $S \subseteq U$ and every $1 \leq i \leq |S|$:

$$\bar{C}(S_i) - \bar{C}(S_{i-1}) \geq 0,$$

where S_i is the set of the first i elements of S (ordered according to $\tau(\cdot, S)$).

Two Crucial Ingredients

Consistent Order Function: for every $S \subseteq T$:

T	1 2 3 4 5 6 7 8 9	$(\tau(\cdot, T)$ order)
S	1 2 3 ● 5 6 8 9	$(\tau(\cdot, T)$ order)
S	1 2 3 9 8 5 6	$(\tau(\cdot, S)$ order)

τ-Increasing: *ALG* is τ-increasing if for every $S \subseteq U$ and every $1 \leq i \leq |S|$:

$$\bar{C}(S_i) - \bar{C}(S_{i-1}) \geq 0,$$

where S_i is the set of the first i elements of S (ordered according to $\tau(\cdot, S)$).

Framework

Let τ be a consistent order function and let *ALG* be a τ-increasing ρ-approximation algorithm for the optimization problem $\mathcal{P}$.

Theorem

The generalized incremental mechanism $M(ALG, \tau)$ is weakly group-strategyproof, ρ-budget balanced and satisfies the no positive transfer property.

[Brenner, Schäfer, SAGT '08]

Our framework reduces the task of designing a WGSP mechanism to finding a consistent order function τ such that the approximation algorithm *ALG* is τ-increasing

Scheduling Example I

Problem: parallel machines, minimize makespan: $P| |C_{max}$

Order Function: order jobs by non-increasing processing times (Graham's rule)

Theorem

The generalized incremental mechanism $M(\text{GRAHAM}, \tau)$ is weakly group-strategyproof and $4/3$-budget balanced.

Contrast: Moulin mechanisms cannot be better than 2-budget balanced

Scheduling Example II

Problem: parallel machines, no preemption, minimize sum of weighted completion times: $P \mid \mid \sum_i w_i C_i$

Order Function: order jobs by non-increasing weight per processing time (Smith's rule)

Theorem

The generalized incremental mechanism $M(\text{SMITH}, \tau)$ is weakly group-strategyproof, 1.21-budget balanced and 2.42-approximate.

Contrast: Moulin mechanisms cannot be better than $\Omega(n)$-budget balanced

Scheduling Example III

Problem: single machine, release dates, preemption, minimize sum of completion times: $1|r_i, pmtn| \sum_i C_i$

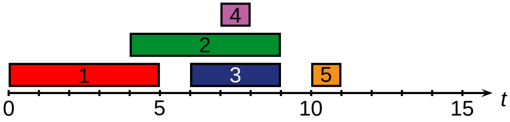
Order Function: order jobs by increasing completion times in the shortest remaining processing time schedule

Theorem

The generalized incremental mechanism $M(\text{SRPT}, \tau)$ is weakly group-strategyproof, 1-budget balanced and 4-approximate.

Contrast: Moulin mechanisms cannot be better than $\Omega(n)$-budget balanced

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

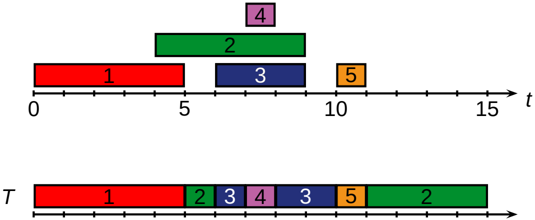
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

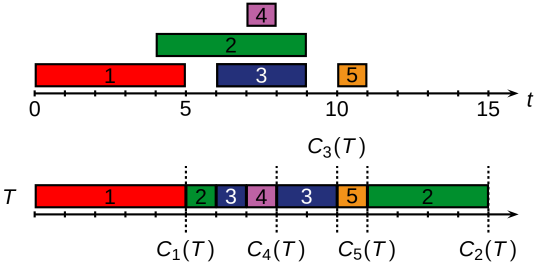
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

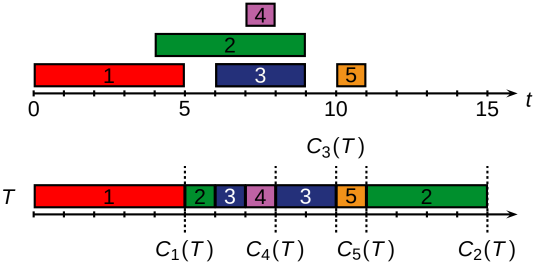
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

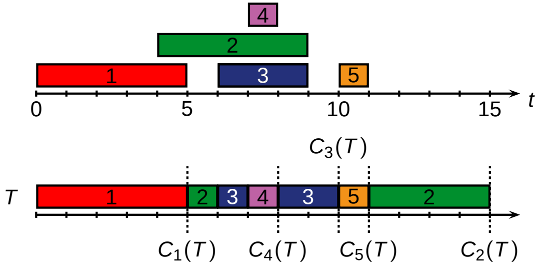
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

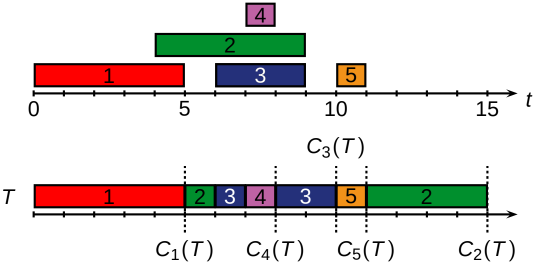
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

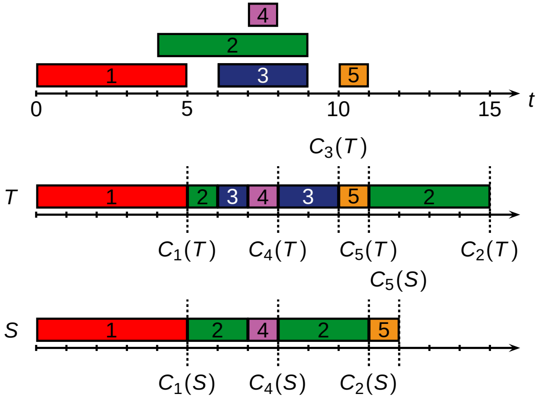
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

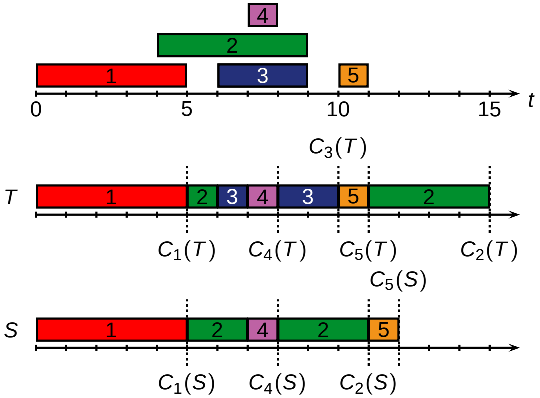
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

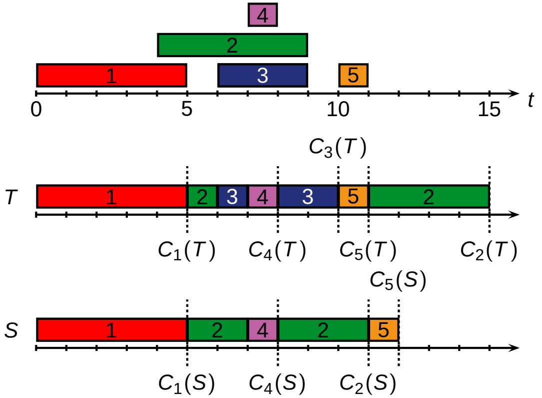
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

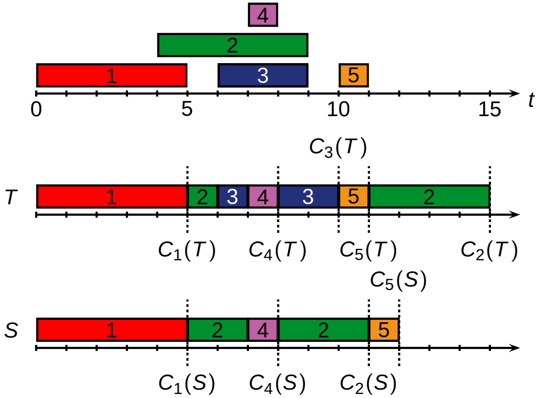
Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

Consistency of SRPT



$T = \{1, \ldots, 5\}$. Suppose we remove Job 3 from T: $S = \{1, 2, 4, 5\}$.

Consider the lifetime of Job 3 in schedule for T:

- Job 2 is a losing job
- Job 4 is a winning job

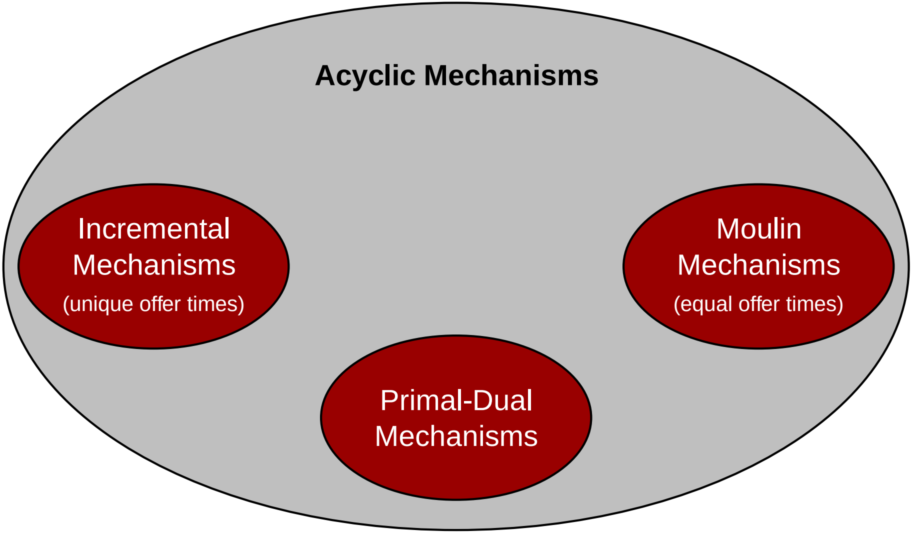
Observation:

- nothing changes for winning jobs
- losing job might be processed in place of Job 3
- but this job will not be completed before $C_3(T)$

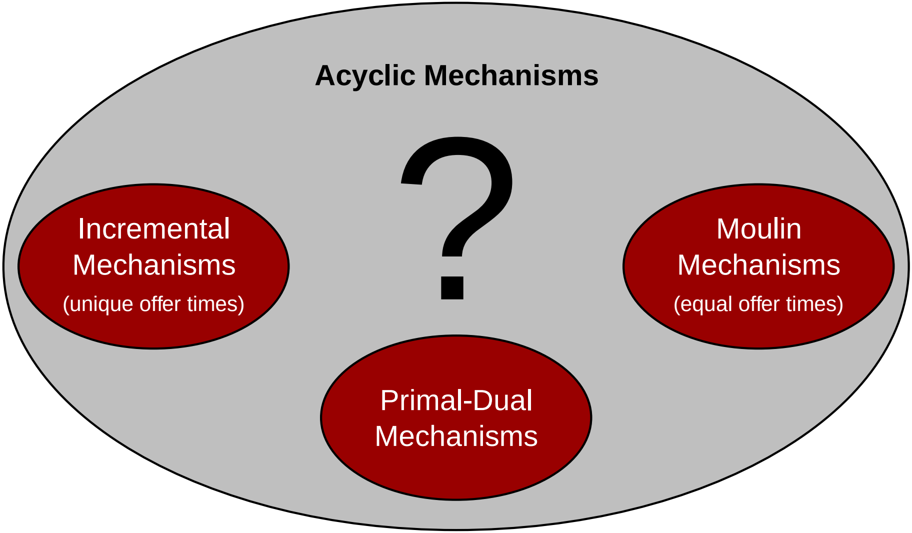
Overview of Results

Problem	our mechanism (β, α)	Moulin mechanism β (lower bound)
$P\|\|C_{\max}$	$\frac{4}{3} - \frac{1}{3m}$	$\frac{2m}{m+1}$
$P\|\|\sum_i C_i$	(1, 2)	$\frac{n+1}{2}$
$P\|\|\sum_i w_i C_i$	(1.21, 2.42)	$\frac{n+1}{2}$
$1\|r_i, pmtn\|\sum_i C_i$	(1, 4)	$\frac{n+1}{2}$
$P\|r_i, pmtn\|\sum_i C_i$	(1.25, 5)	$\frac{n+1}{2}$
$1\|r_i, pmtn\|\sum_i F_i$	1	$\frac{n+1}{2}$
MST	1	1
Steiner tree	2	2
TSP	2	–

Universe of Acyclic Mechanisms



Universe of Acyclic Mechanisms



Conclusions and Open Problems

Conclusions

Moulin Mechanisms:

- achieve strong notion of group-strategyproofness
- only known framework to derive GSP mechanisms
- may suffer from bad budget balance or social cost approximation factors
- cross-monotonic cost shares derived in case-by-case studies

Our Framework:

- weaker notion of weakly group-strategyproofness, but coalitional viewpoint retained
- framework to derive WGSP mechanisms from existing algorithms, thereby preserving approximation factor
- yields constant budget balance and social cost approximation guarantees, e.g., for scheduling problems

Open Problems

Open Problem: Which other algorithms exploit the full strength of our framework? Which types of algorithms satisfy consistency?

Open Problem: Are there other approaches to derive acyclic mechanisms from approximation algorithms?

Open Problem: What are the trade-offs between weakly group-strategyproofness and budget balance and social cost approximation guarantees?

Open Problem: Consider more general settings such as online, general demand, etc. (see also [Brenner, Schäfer, CIAC '10])

Approximation Algorithms for Rent-or-Buy Problems

Multicommodity Rent-or-Buy

Given:

- graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}^+$
- set of k terminal pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\}$
- demand d_i for commodity (s_i, t_i)
- parameter $M \geq 1$

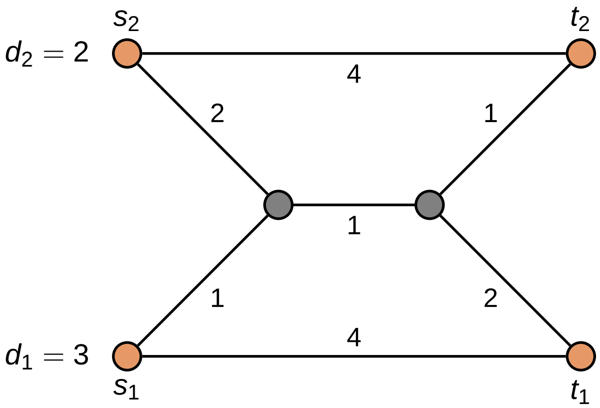
Rent-or-Buy: on each edge e:

- either rent capacity $\lambda(e)$ at cost $\lambda(e) \cdot c_e$
- or buy infinite capacity at cost $M \cdot c_e$

Goal: determine minimum-cost capacity installation such that all demands can be routed simultaneously

Example: Multicommodity Rent-or-Buy

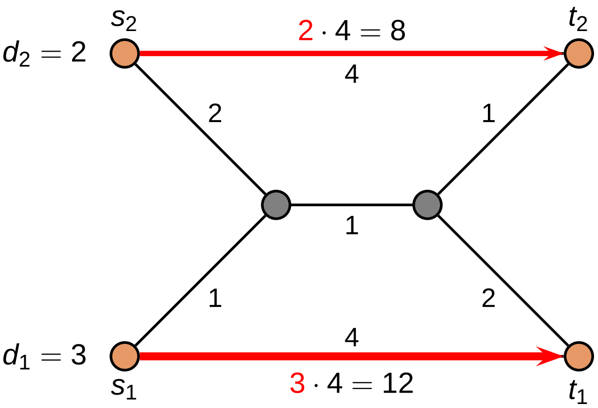
$M = 4$



Example: Multicommodity Rent-or-Buy

$M = 4$

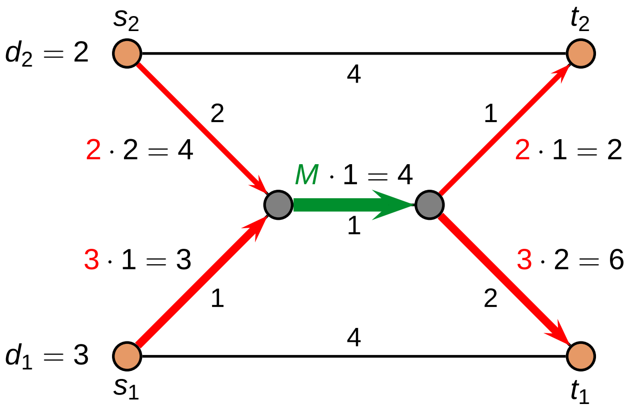
capacity installation cost: 20



Example: Multicommodity Rent-or-Buy

$M = 4$

capacity installation cost: 19



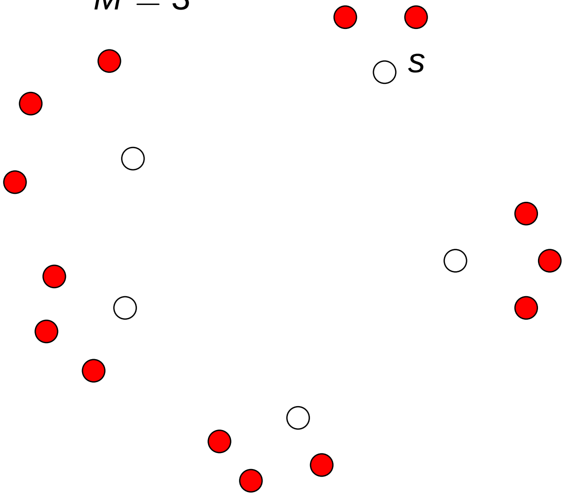
Special Cases

Steiner Forest (unit demands, $M = 1$):
Given a graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}^+$ and k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$, find a minimum-cost forest F in G that contains an s_i, t_i-path for all i.

Single-Sink Rent-or-Buy:
Same input as for MROB, but all terminal pairs share a common sink node s.

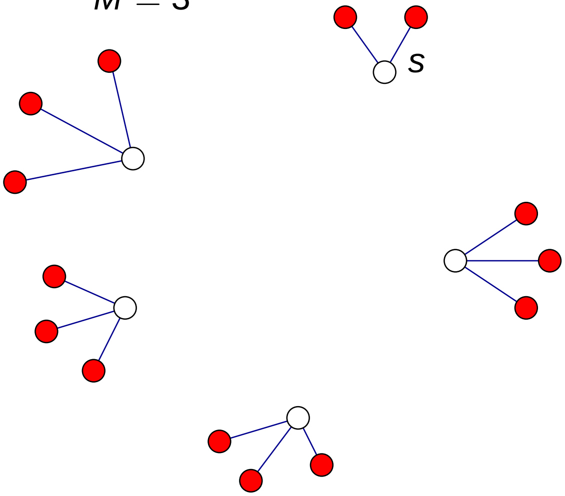
Example: Single-Sink Rent-or-Buy

$M = 3$

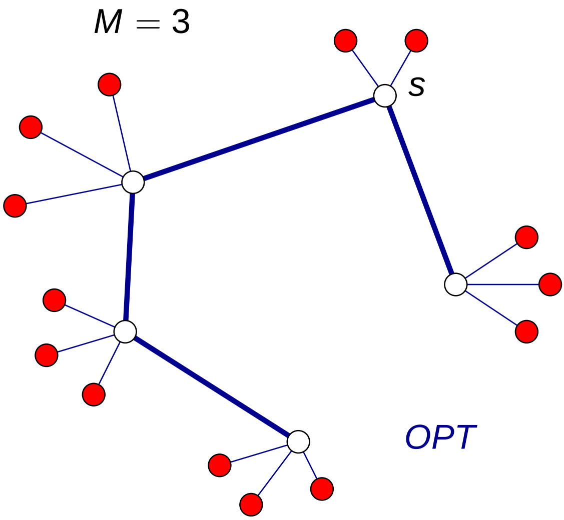


Example: Single-Sink Rent-or-Buy

$M = 3$



Example: Single-Sink Rent-or-Buy



Connected Facility Location*

Given:

- graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}^+$
- set $D \subseteq V$ of demands
- parameter $M \geq 1$

Goal:

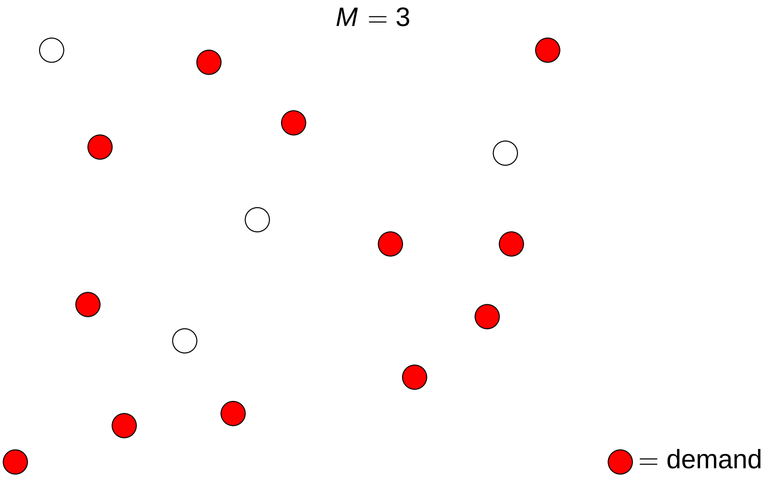
- find a subset $F \subseteq V$ of facilities that are opened
- connect each $j \in D$ to some open facility $\sigma(j) \in F$
- build a Steiner tree T on F so as to minimize

$$M \cdot c(T) + \sum_{j \in D} \ell(j, \sigma(j))$$

$\ell(u, v) =$ shortest path distance between nodes u and v in G

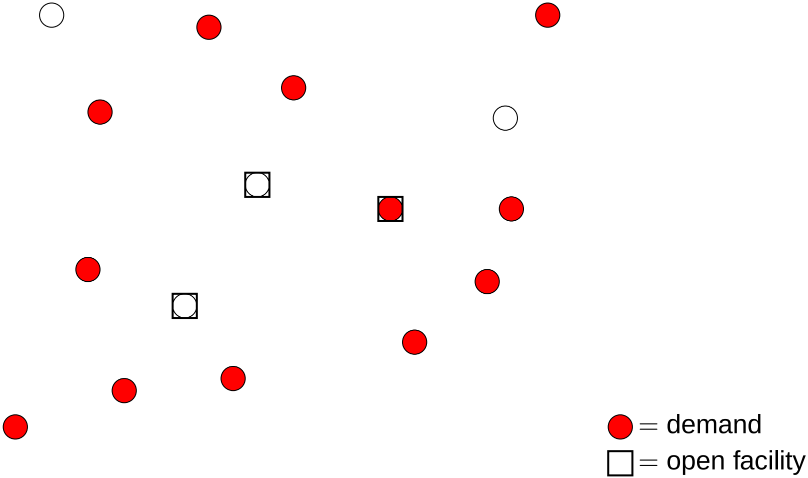
*__Note:__ every node is a facility and there are no opening costs

Example: Connected Facility Location*

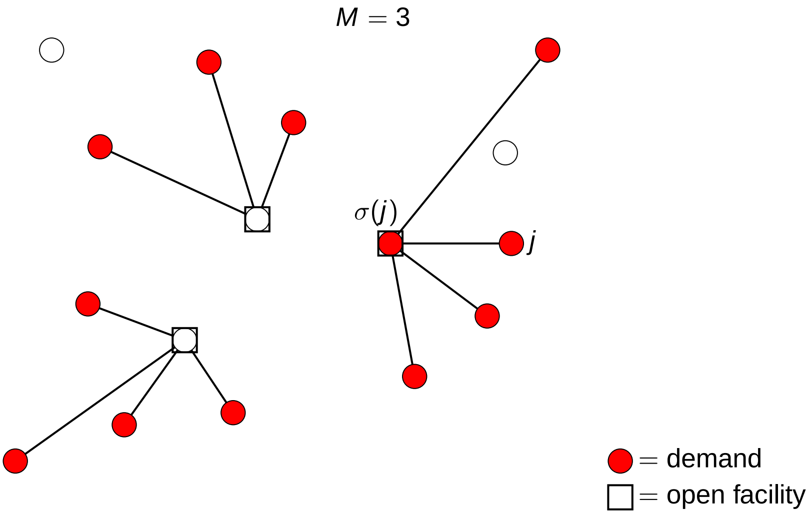


Example: Connected Facility Location*

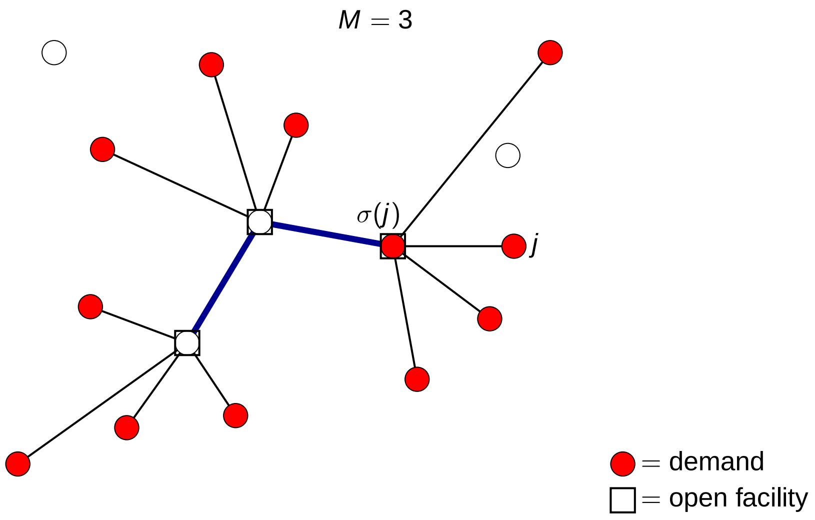
$M = 3$



Example: Connected Facility Location*



Example: Connected Facility Location*



Randomized Framework

Assumption: can assume without loss of generality that every terminal pair has unit demand

Sample-and-Augment Algorithm for MROB:

1: Mark each terminal pair with probability $1/M$. Let D be set of marked terminal pairs.
2: Compute an α-approximate Steiner forest F for D and buy all edges in F.
3: For all terminal pairs $(s, t) \notin D$: rent unit capacity on a shortest s, t-path in contracted graph $G|F$.

$G|F$ = graph obtained from G by contracting all edges in $F \subseteq E$

Strictness Concept

Definition

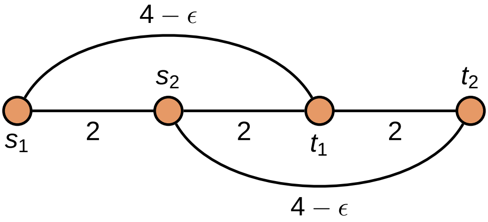
A Steiner forest algorithm *ALG* is β-strict if there exist **cost shares** $\xi_{st} \geq 0$ for every $(s,t) \in R$ such that:

1. $\sum_{(s,t)\in R} \xi_{st} \leq c(F^*)$ (competitiveness)
2. For every $(s,t) \in R$, $c_{G|F_{-st}}(s,t) \leq \beta \cdot \xi_{st}$ (β-strictness)

Notation:

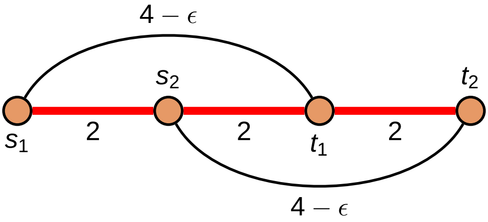
- $F^* =$ optimal Steiner forest for R
- $F_{-st} =$ Steiner forest computed by *ALG* for $R_{-st} = R \setminus \{(s,t)\}$
- $G|F_{-st} =$ graph obtained if all components of F_{-st} are contracted

Example: Strictness

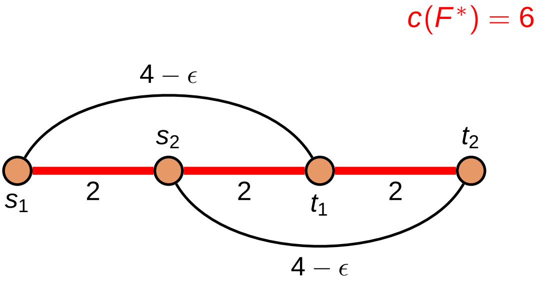


Example: Strictness

$c(F^*) = 6$

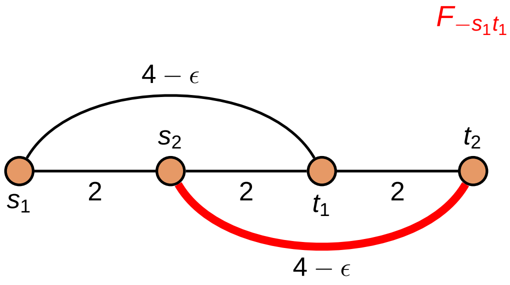


Example: Strictness



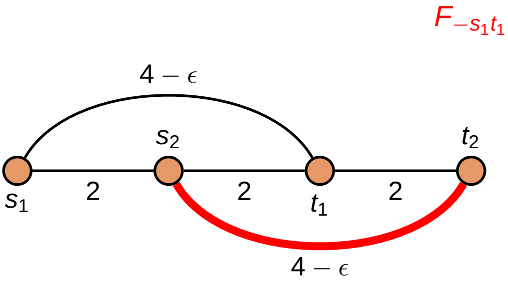
Suppose: $\xi_{s_1 t_1} = \xi_{s_2 t_2} = 3$

Example: Strictness



Suppose: $\xi_{s_1 t_1} = \xi_{s_2 t_2} = 3$

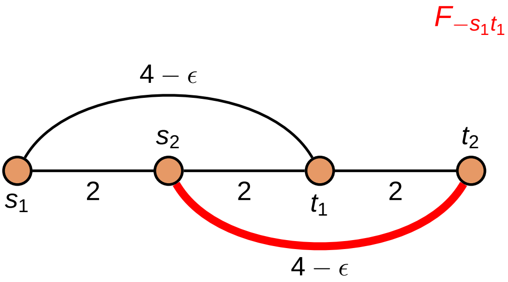
Example: Strictness



Suppose: $\xi_{s_1 t_1} = \xi_{s_2 t_2} = 3$

$c_{G|F_{-s_1 t_1}}(s_1, t_1) = 4 - \epsilon$

Example: Strictness

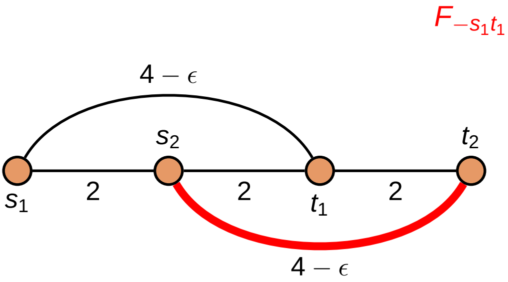


Suppose: $\xi_{s_1 t_1} = \xi_{s_2 t_2} = 3$

$c_{G|F_{-s_1 t_1}}(s_1, t_1) = 4 - \epsilon$

$\frac{4}{3} \cdot \xi_{s_1 t_1}$ sufficient to connect s_1 and t_1 in $G|F_{-s_1 t_1}$

Example: Strictness



Suppose: $\xi_{s_1t_1} = \xi_{s_2t_2} = 3$

$c_{G|F_{-s_1t_1}}(s_1, t_1) = 4 - \epsilon$

$\frac{4}{3} \cdot \xi_{s_1t_1}$ sufficient to connect s_1 and t_1 in $G|F_{-s_1t_1}$

similar for $(s_2, t_2) \Rightarrow \frac{4}{3}$-strict

Randomized Framework

Theorem

Given an α-approximate and β-strict Steiner forest algorithm, Sample-and-Augment is an (expected) $(\alpha + \beta)$-approximation algorithm for MROB.

[Gupta, Kumar, Pál, Roughgarden, JACM '07]

Remark: framework applies to other network design problems

- single-sink rent-or-buy
- multicast rent-or-buy
- virtual private network design
- single-sink buy-at-bulk