GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Seminar Approximation Algorithms

Approximating Nash Social Welfare under Submodular Valuations through (Un)Matchings

Based on a paper of the same title by J. Garg, P. Kulkarni, and R. Kulkarni

Zeno Adrian Weil
Supervised by Dr Giovanna Varricchio

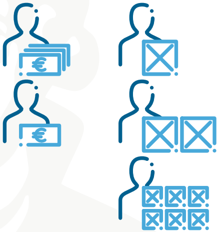
19th July 2023 · Algorithms and Complexity (Prof. Dr Martin Hoefer)

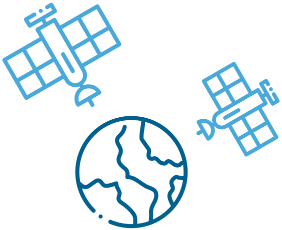
What is the issue?

We need to distribute goods amongst recipients *efficiently* and *fairly*.

Where is this encountered?

- industrial procurement
- satellites
- water withdrawal





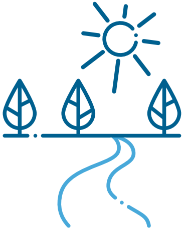


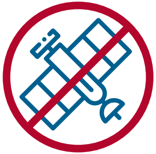
Table of Contents

1 Preliminaries

Allocations

Setting:

- recipients: set $\mathscr{A}$ of n agents
- goods: set $\mathscr{G}$ of m items
  - unsharable
  - indivisible



Definition

An *allocation* is a tuple $\boldsymbol{x} = (\boldsymbol{x}_i)_{i \in \mathscr{A}}$ of bundles $\boldsymbol{x}_i \subset \mathscr{G}$ such that each item is element of precisely one bundle.
Item j is *assigned* to agent i if $j \in \boldsymbol{x}_i$.

But how to measure its efficiency and fairness?

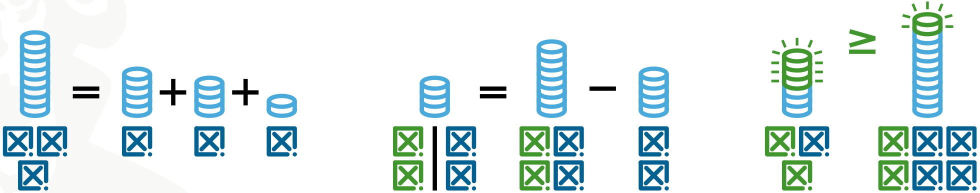
Valuation Functions

Requirements:

- monotonically non-decreasing: $v_i(\mathcal{S}_1) \leq v_i(\mathcal{S}_2)$
- normalised: $v_i(\emptyset) = 0$

Types:

- additive: $v_i(\mathcal{S}) := \sum_{j \in \mathcal{S}} v_i(j)$
- submodular: $v_i(\mathcal{S}_1 \mid \mathcal{S}_2) := v_i(\mathcal{S}_1 \cup \mathcal{S}_2) - v_i(\mathcal{S}_2)$
  - diminishing returns



Asymmetric Maximum Nash Social Welfare Problem

Problem

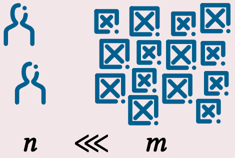
$$x^* \overset{!}{=} \underset{x \in X_{\mathscr{A}}(\mathscr{G})}{\arg\max}\{\mathrm{NSW}(x)\} \quad \text{with } \mathrm{NSW}(x) := \Big(\prod_{i \in \mathscr{A}} v_i(x_i)^{\eta_i}\Big)^{1/\sum_{i \in \mathscr{A}} \eta_i}$$

- $X_{\mathscr{A}}(\mathscr{G})$: all possible allocations
- η_i: agent weight

The NSW strikes a middle ground between efficiency and fairness!

Is there a polynomial-time algorithm with an approximation factor ...

- ... dependent on n?
- ... independent from m?



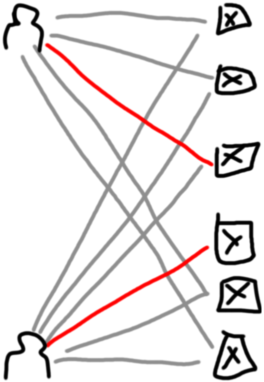
2

RepReMatch

Naïve Approach

Naïve approach:

- repeatedly use maximum matchings
- fails because of missing foresight
  - additive valuations: sort items by valuation
    $\Rightarrow 2n$-approximation (SMatch)
  - submodular valuations: set of lowest valuation
    approximable only by $\Omega(\sqrt{m/\ln m})$ ⚡



Key Ideas of the Algorithm

We need change the past in three phases:

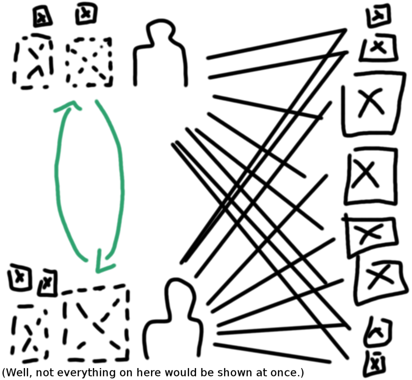
Phase I Assign enough high-value items temporarily.

Phase II Assign the remaining items definitely.

Phase III Re-assign the items of phase I definitely.

Theorem

RepReMatch guarantees a $2n(\log_2 n + 3)$-approximation under submodular valuations.



(Well, not everything on here would be shown at once.)

The Algorithm

Phase I:

1. repeat $\lceil \log_2 n \rceil + 1$ times or until $\mathcal{G} = \emptyset$
    1. create bipartite graph $(\mathcal{A}, \mathcal{G}, E)$ with edge weights $\eta_i \cdot \log v_i(j)$
    2. compute maximum weight matching
    3. update bundles $\boldsymbol{x}_i^{\mathrm{I}}$ & remove assigned items

Phase II:

2. repeat until $\mathcal{G} = \emptyset$
    1. create bipartite graph $(\mathcal{A}, \mathcal{G}, E)$ with edge weights $\eta_i \cdot \log v_i(\boldsymbol{x}_i^{\mathrm{II}} \cup \{j\})$
    2. compute maximum weight matching
    3. update bundles $\boldsymbol{x}_i^{\mathrm{II}}$ & remove assigned items

Phase III:

3. create bipartite graph $\left(\mathcal{A}, \bigcup_{i \in \mathcal{A}} \boldsymbol{x}_i^{\mathrm{I}}, E\right)$ with edge weights $\eta_i \cdot \log v_i(\boldsymbol{x}_i^{\mathrm{II}} \cup \{j\})$
4. compute maximum weight matching
5. create bundles $\boldsymbol{x}_i^{\mathrm{III}}$

Analysing Phases I & III (1/2)

Phase I reserves ‘high-value’ items. But what qualifies as ‘high-value’?

Definition

Let $\boldsymbol{x}_i^* = \{o_i^1, o_i^2, \dots\}$ be an optimal bundle. An item $j \in \mathcal{G}$ is *outstanding* if $v_i(j) \geq v_i(o_i^1)$.

$\Rightarrow$ Are enough outstanding items reserved?

Analysing Phases I & III (2/2)

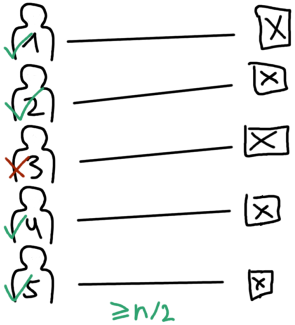
Lemma

Each agent can be matched with an outstanding item in phase III.

- maximum number of unmatched agents halved with each round of phase I
  - $\lceil \log_2 n \rceil + 1$ rounds in phase I are enough
- induction on number of rounds in phase I

Base Case: In round 1 of phase I, either

- $\geq n/2$ many agents matched with an outstanding item
- $< n/2$ many agents matched with an outstanding item
  - $> n/2$ many items o_i^1 assigned to someone else
  - $> n/2$ many agents matched upon release in phase III



Analysing Phases I & III (2/2)

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

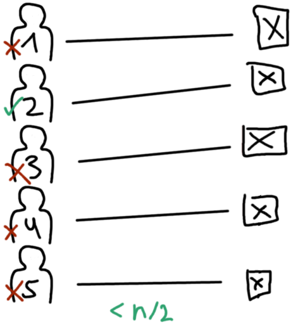
Lemma

Each agent can be matched with an outstanding item in phase III.

- maximum number of unmatched agents halved with each round of phase I
  - $\lceil \log_2 n \rceil + 1$ rounds in phase I are enough
- induction on number of rounds in phase I

Base Case: In round 1 of phase I, either

- $\geq n/2$ many agents matched with an outstanding item
- $< n/2$ many agents matched with an outstanding item
  - $> n/2$ many items o_i^1 assigned to someone else
  - $> n/2$ many agents matched upon release in phase III



Analysing Phases I & III (2/2)

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

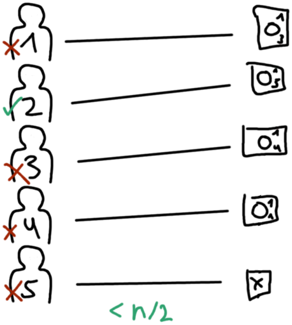
Lemma

Each agent can be matched with an outstanding item in phase III.

- maximum number of unmatched agents halved with each round of phase I
  - $\lceil \log_2 n \rceil + 1$ rounds in phase I are enough
- induction on number of rounds in phase I

Base Case: In round 1 of phase I, either

- $\geq n/2$ many agents matched with an outstanding item
- $< n/2$ many agents matched with an outstanding item
  - $> n/2$ many items o_i^1 assigned to someone else
  - $> n/2$ many agents matched upon release in phase III



Analysing Phases I & III (2/2)

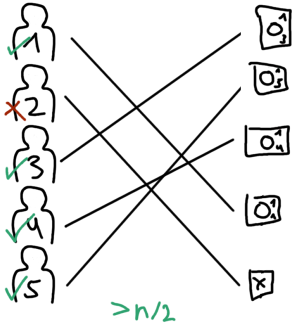
Lemma

Each agent can be matched with an outstanding item in phase III.

- maximum number of unmatched agents halved with each round of phase I
  - $\lceil \log_2 n \rceil + 1$ rounds in phase I are enough
- induction on number of rounds in phase I

Base Case: In round 1 of phase I, either

- $\geq n/2$ many agents matched with an outstanding item
- $< n/2$ many agents matched with an outstanding item
  - $> n/2$ many items o_i^1 assigned to someone else
  - $> n/2$ many agents matched upon release in phase III



Analysing Phase II (1/2)

Definition

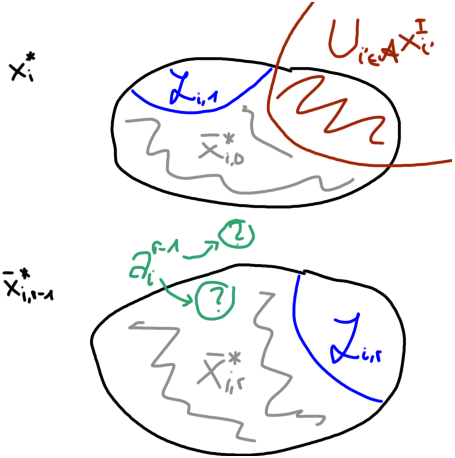
The set $\mathscr{L}_{i,r}$ of *lost items* is the set of all optimal items $j \in \boldsymbol{x}_i^*$ assigned to other agents $i' \neq i$ in round r.

Definition

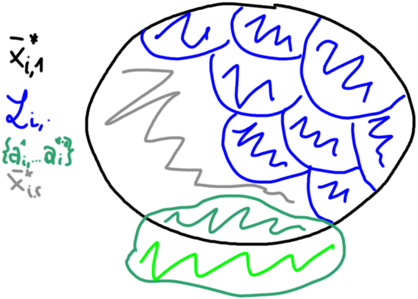
Let $\boldsymbol{x}_i^{\mathrm{II}} = \{a_i^1, a_i^2, \dots\}$ be the bundle of agent i. The set of *optimal and attainable items* is defined as

$$\bar{\boldsymbol{x}}_{i,r}^* := \begin{cases} \boldsymbol{x}_i^* \setminus \left(\bigcup_{i' \in \mathscr{A}} \boldsymbol{x}_{i'}^{\mathrm{I}} \cup \mathscr{L}_{i,1}\right) & \text{in round } r = 1, \\ \bar{\boldsymbol{x}}_{i,r-1}^* \setminus (\mathscr{L}_{i,r} \cup \{a_i^{r-1}\}) & \text{in round } r \geq 2. \end{cases}$$

⇒ What is the valuation of the remaining items?



Analysing Phase II (2/2)



maybe auxiliary calculation for
$v_i(\mathscr{L}_{i,l} \mid a_i^1, \dots, a_i^{l-2})$
$= |\mathscr{L}_{i,l}| \cdot v_i(a_i^{l-1} \mid a_i^1, \dots, a_i^{l-2})$
here

$$v_i(\overline{\boldsymbol{x}}^*_{i,r} \mid a_i^1, \dots, a_i^{r-1}) \geq v_i(\overline{\boldsymbol{x}}^*_{i,r}) - v_i(a_i^1, \dots, a_i^{r-1}) - \sum_{l=2}^{r} |\mathscr{L}_{i,l}| \cdot v_i(a_i^{l-1} \mid a_i^1, \dots, a_i^{l-2})$$

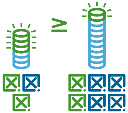
Plan: first show black set, then alternately enlarge green set and uncover blue sets $\Rightarrow$ valuation of grey area = val. of black – val. of dark green – val. of blue $\Rightarrow$ lower bound is enough, therefore subtract val. of whole green area $\Rightarrow$ show that sum of marg. val. of a_i^l equals val. of $a_i^1, \dots, a_i^{r-1}$ $\Rightarrow$ then subtract marg. val. of blue area by summing over marg. val. of each lost set $\Rightarrow$ marg. val. of lost set $\leq$ sum of marg. val. of items of lost set $\Rightarrow$ marg. val. of item of lost set $\leq$ marg. val. of a_i^{l-1} because a_i^{l-1} assigned before items in lost set

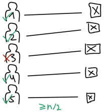
Conclusion

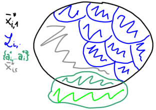
Summary & Outlook

- allocation: partition of items amongst agents
- bundles valued using submodular valuation functions
  - diminishing returns
- Nash social welfare: weighted geometric mean of valuations
- approximation factor independent from m?
- simple, repeated matching fails because of missing foresight
- RepReMatch: $2n(\log n + 3)$-approximative

Phase I finding enough outstanding items
Phase II assigning remaining item
Phase III assigning outstanding items







Any Room for Improvement?

Possibly! Lower bound of 1.72.

End of Talk