Seminar Approximation Algorithms

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

Based on a paper of the same name by Garg, Kulkarni and Kulkarni

Zeno Adrian Weil
Supervised by Dr Giovanna Varricchio

15th July 2023 · Algorithms and Complexity (Prof. Dr Martin Hoefer)

What is the issue?

We need to distribute goods amongst recipients *fast*, *efficient* and *fairly*.

Where is this encountered?

- industrial procurement
- cloud services
- satellites
- water withdrawal

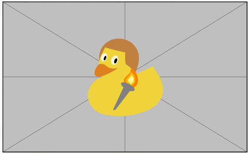
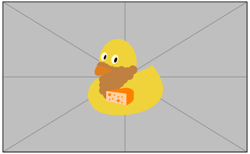
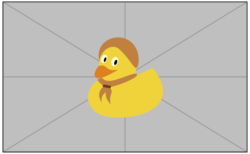
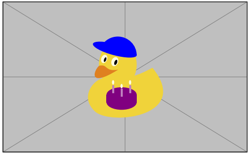





Table of Contents

Preliminaries

Allocations

Setting:

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

Definition (1)

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?

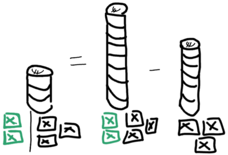
Valuation Functions

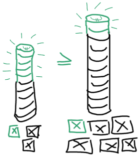
Requirements:

- monotonically non-decreasing: $v_i(\mathcal{S}_1) \leq v_i(\mathcal{S}_2) \quad \forall \mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{G}$
- normalised: $v_i(\emptyset) = 0$
- non-negative: $v_i(\mathcal{S}) \geq 0 \quad \forall \mathcal{S} \subset \mathcal{G}$

Types:

- additive: $v_i(\mathcal{S}) := \sum_{j \in \mathcal{S}} v_i(j) \quad \forall \mathcal{S} \subset \mathcal{G}$
- submodular: $v_i(\mathcal{S}_1 \mid \mathcal{S}_2) := v_i(\mathcal{S}_1 \cup \mathcal{S}_2) - v_i(\mathcal{S}_2) \quad \forall \mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{G}$ with $\mathcal{S}_1, \mathcal{S}_2$ disjoint
  - more general (encompasses additivity)
  - diminishing returns





Asymmetric Maximum Nash Social Welfare Problem

Problem (2)

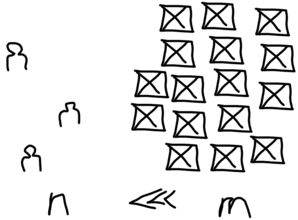
$$x^* \stackrel{!}{=} \operatorname*{arg\,max}_{x \in X_{\mathscr{A}}(\mathscr{G})} \{\mathrm{NSW}(x)\} \quad \text{with } \mathrm{NSW}(x) := \Big(\prod_{i \in \mathscr{A}} v_i(x_i)^{\eta_i}\Big)^{1/\sum_{i\in\mathscr{A}} \eta_i}$$

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

The NSW strikes a middle ground between efficiency and fairness!

Challenge

Algorithm with approximation factor *independent from m*!

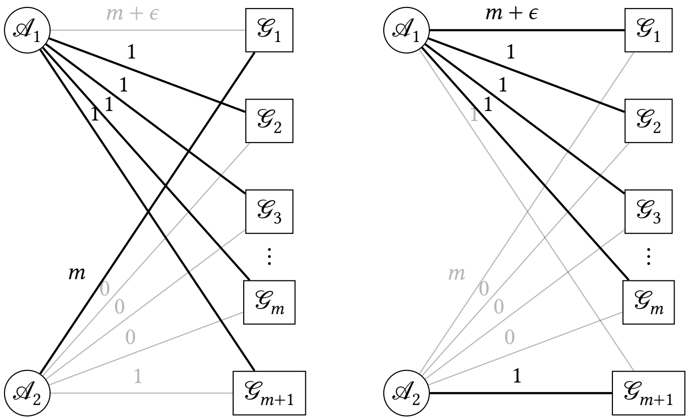


2

RepReMatch

Naïve Approach

$$|\mathscr{A}| = 2$$
$$|\mathscr{G}| = m + 1$$
$$\eta_1 = \eta_2 = 1$$
$$\begin{aligned}\mathrm{NSW}(\boldsymbol{x}_i^*) &= \sqrt{m \cdot m} \\ &= m\end{aligned}$$
$$\begin{aligned}\mathrm{NSW}(\boldsymbol{x}_i) &= \sqrt{(2m + \epsilon - 1) \cdot 1} \\ &\leq \sqrt{2m}\end{aligned}$$
$$\Rightarrow \alpha \geq \sqrt{m/2}$$



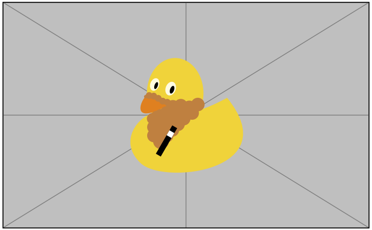
What are the items of lowest valuation?

Key Ideas of the Algorithm

Under submodular valuations, the set of lowest valuation is approximable only by $\Omega\left(\sqrt{m/\ln m}\right)$. $\lightning$

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.



A $2n(\log_2 n + 3)$-approximation is possible!

The Algorithm

Phase I

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

Phase II

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

Phase III

3. create bipartite graph $G = (\mathscr{A}, \bigcup_{i \in \mathscr{A}} \boldsymbol{x}_i^{\mathrm{I}}, E)$ with edge weights $w(i, j) = \eta_i \log\big(v_i(\boldsymbol{x}_i^{\mathrm{II}} \cup \{j\})\big)$
4. compute maximum weight matching $\mathscr{M}$
5. create bundles $\boldsymbol{x}_i^{\mathrm{III}}$ according to matching $\mathscr{M}$ and previous bundles $\boldsymbol{x}_i^{\mathrm{II}}$

Analysing Phases I & III (1/2)

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

Definition (14)

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 (15)

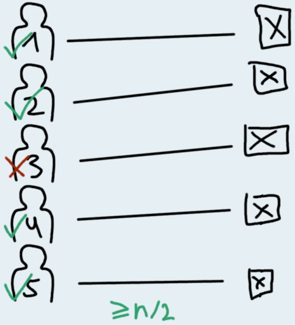
Each agent can be matched with an outstanding item in phase III.

Sketch Proof

- 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 (15)

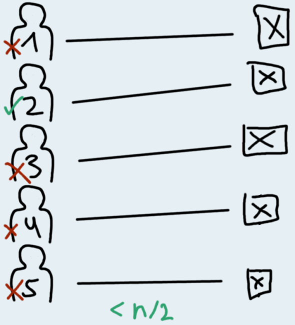
Each agent can be matched with an outstanding item in phase III.

Sketch Proof

- 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 (15)

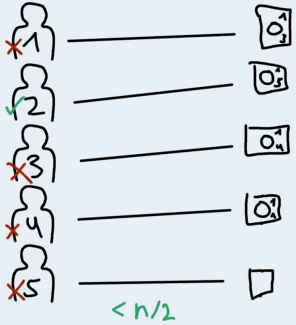
Each agent can be matched with an outstanding item in phase III.

Sketch Proof

- 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



$\square$

Analysing Phases I & III (2/2)

Lemma (15)

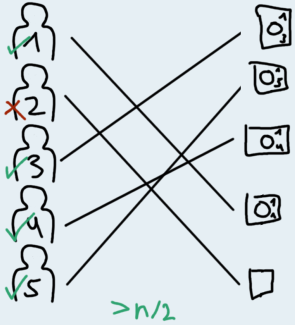
Each agent can be matched with an outstanding item in phase III.

Sketch Proof

- 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 (9)

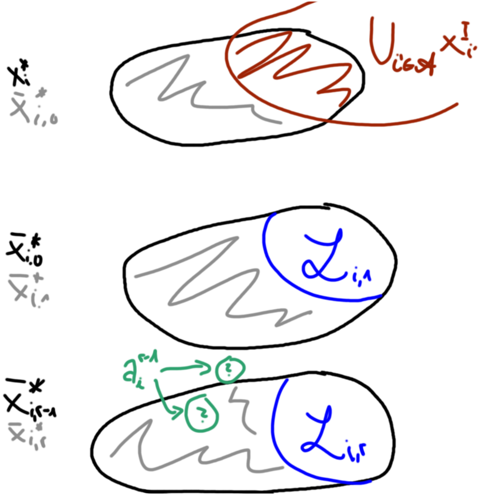
The set $\mathscr{L}_{i,r}$ of *lost items* is the set of all optimal items $j \in \boldsymbol{x}_i^*$ assigned to other agents $i' \neq i$ in round r.

Definition (10)

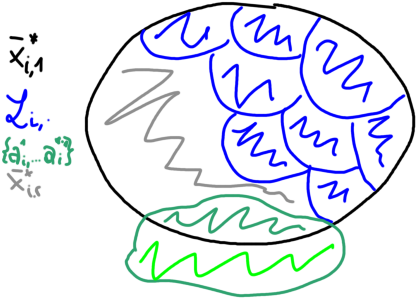
Let $\boldsymbol{x}_i^{\mathrm{II}} = \{a_i^1, a_i^2, \dots\}$ be the bundle of agent i. The set of *optimal and attainable items* is defined as

$$\overline{\boldsymbol{x}}_{i,r}^* := \begin{cases} \boldsymbol{x}_i^* \setminus \bigcup_{i' \in \mathscr{A}} \boldsymbol{x}_{i'}^{\mathrm{I}} & \text{in round } r = 0, \\ \overline{\boldsymbol{x}}_{i,0}^* \setminus \mathscr{L}_{i,1} & \text{in round } r = 1, \\ \overline{\boldsymbol{x}}_{i,r-1}^* \setminus (\{a_i^{r-1} \cup \mathscr{L}_{i,r})\}) & \text{in round } r \geq 2. \end{cases}$$

⇒ What is the valuation of the remaining items?



Analysing Phase II (2/2)



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

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

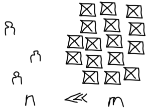
Plan: first show black set, then alternately enlarge green set and uncover blue sets ⇒ valuation of grey area = val. of black – val. of dark green – val. of blue ⇒ lower bound is enough, therefore subtract val. of whole green area ⇒ show that sum of marg. val. of a_i^l equals val. of $a_i^1, \ldots,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

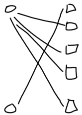
Conclusion

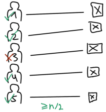
Summary

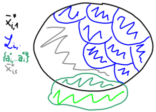
- allocation: partition of items amongst agents
- bundles valued using submodular valuation functions
  - diminishing returns
- Nash social welfare: weighted geometric mean of valuations
- approximation factor independent from m?
- simple, repeated matching fails because of missing foresight
- RepReMatch: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

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









Outlook

An improvement over previous results?
Yes! $(m - n + 1)$ best known approximation factor before.

Any room for improvement left?
Possibly! Lower bound of 1.72.

Other approaches:

- less general valuation functions $\rightsquigarrow$ better factors
- limits on agent weights $\rightsquigarrow$ linear and even constant factors

End of Talk