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

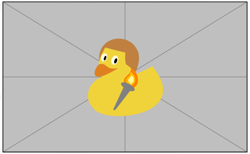
8th July 2023 · Algorithms and Complexity (Prof. Dr Martin Hoefer)

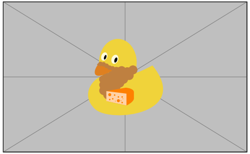
What is the issue?

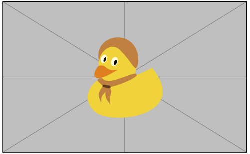
We need to distribute goods amongst recipients *fast*, *efficient* and *fairly*.

Where is this encountered?

- industrial procurement
- mobile edge computing
- satellites
- water withdrawal







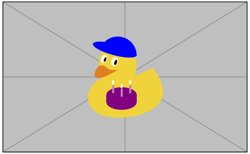


Table of Contents

Preliminaries

Allocations

Setting:

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

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?

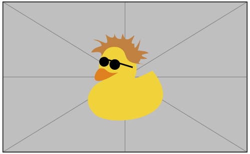
Valuation Functions

Requirements:

- monotonically non-decreasing: $v_i(\mathcal{S}_1) \leq v_i(\mathcal{S}_2) \quad \forall \mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{G}$
- 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

Definition

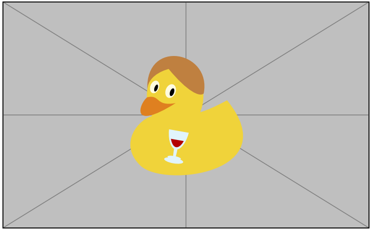
$$\boldsymbol{x}^* \stackrel{!}{=} \underset{\boldsymbol{x} \in X_{\mathscr{A}}(\mathscr{G})}{\arg\max} \{\mathrm{NSW}(\boldsymbol{x})\} \quad \text{with } \mathrm{NSW}(\boldsymbol{x}) := \Big(\prod_{i \in \mathscr{A}} v_i(\boldsymbol{x}_i)^{\eta_i}\Big)^{1/\sum_{i \in \mathscr{A}} \eta_i}$$

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

- middle ground between efficiency and fairness

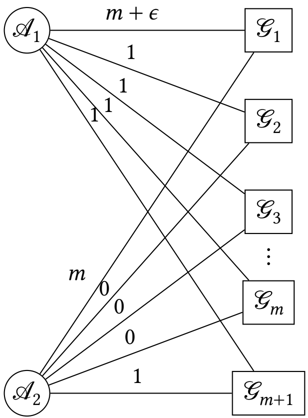
Challenge

Algorithm with approximation factor *independent from m*!



RepReMatch

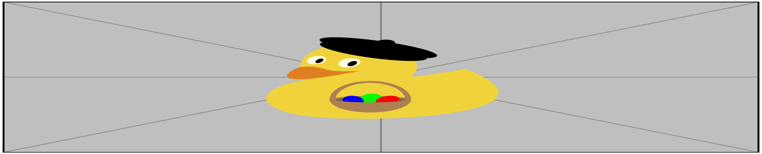
Naïve Approach



What are the low-value items?

Looking into the Future

- sort items by valuation in descending order
  - low-value items on the left



- use their valuations for edge weights in early matchings

A $2n$-approximation is possible … using SMatch.

This only works for additive valuation functions.

Changing the Past

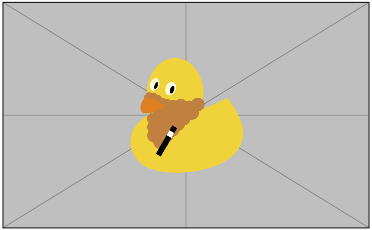
Under submodular valuation, the set of lowest valuation is approximable only by $\Omega\left(\sqrt{m/\ln m}\right)$.

We can 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 $\mathscr{M}$ & 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 $\mathscr{M}$ & 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 Phase II (1/3)

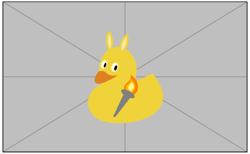
Definition

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

Definition

The set of *optimal and attainable items* is defined as

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



Analysing Phase II (2/3)

Lemma

For each agent $i \in \mathscr{A}$ and her bundle $\boldsymbol{x}_i^{\mathrm{II}} = \{a_i^1, \dots, a_i^{\tau_i^{\mathrm{II}}}\}$ at the end of phase II, it holds in all rounds $r = 2, \dots, \tau_i^{\mathrm{II}}$ that

$$v_i(\overline{\boldsymbol{x}}_{i,r}^* \mid a_i^1, \dots, a_i^{r-1}) \geq v_i(\overline{\boldsymbol{x}}_{i,1}^*) - \sum_{r'=1}^{r-1} \ell_{i,r'+1} \cdot v_i(a_i^{r'} \mid a_i^1, \dots, a_i^{r'-1}) - v_i(a_i^1, \dots, a_i^{r-1}) .$$