Algorithms and Complexity Group
Institute for Computer Science
Goethe University Frankfurt

Summer Semester 2023

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 [13].

Żeno Adrian Weil

24th June 2023

Examiner: Prof Dr Martin Hoefer
Supervisor: Dr Giovanna Varricchio

Abstract

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Contents

Todo list

1 Introduction

1.1 Motivation

The study of distributing goods amongst several receivers is an interdisciplinary field, pursued as early as the 1940s [19]. It is interesting both computationally — how to distribute fast — and qualitatively — how to distribute well. Its areas of application are manifold:

- For industrial procurement, the preferences of buyers and sellers need be appropriately captured and real-world constraints on goods and services be taken into account [7].
- Mobile Edge Computing describes a technique where mobile devices compete for computational and storage capabilities of physically close servers. However, the participation of users and the provision of the serves has to be incentivised and monetised [2, 10].
- Many production sites, machines, etc. are required for manufacturing, and the tasks between them must be scheduled efficiently. Disturbances must be quickly paid heed to [7].
- Water is a crucial resource, and even hostile countries must come to mutual agreements on the withdrawal from contested rivers [9].

1.2 Preliminaries

In this seminar paper, we focus on unsharable and indivisible goods, which we term *items*. The receivers of those items are termed *agents*. The distributions of items amongst agents are modelled through allocations.

Definition 1. Let $\mathcal{G}$ be a set of m items and $\mathcal{A}$ be a set of n agents. An *allocation* is a tuple $\boldsymbol{x} = (\boldsymbol{x}_i)_{i \in \mathcal{A}}$ of *bundles* $\boldsymbol{x}_i \subset \mathcal{G}$ such that each item is element of exactly one bundle, that is, $\bigcup_{i \in \mathcal{A}} \boldsymbol{x}_i = \mathcal{G}$ and $\boldsymbol{x}_i \cap \boldsymbol{x}_{i'} = \emptyset$ for all $i \neq i'$. An item $j \in \mathcal{G}$ is *assigned* to agent $i \in \mathcal{A}$ if $j \in \boldsymbol{x}_i$ holds.

or rather 'allocated'?

The satisfaction of an agent i with her bundle $\boldsymbol{x}_i$ is measured by her *valuation function* v_i, which assigns each set of items a real value. We always assume that valuation functions are monotonically non-decreasing, i. e., $v_i(\mathcal{S}_1) \leq v_i(\mathcal{S}_2)\ \forall \mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{G}$, and normalised, i. e., $v_i(\emptyset) = 0$. Note that this implies non-negativity, i. e., $v_i(\mathcal{S}) \geq 0\ \forall \mathcal{S} \subset \mathcal{G}$. Besides fulfilling these properties, the valuation functions can come from a plethora of function families. We discuss additive and submodular functions in greater detail.

Additive The valuation $v_i(\mathcal{S})$ of an agent $i \in \mathcal{A}$ for any set $\mathcal{S} \subset \mathcal{G}$ of items is the sum of the valuations of the individual items $j \in \mathcal{S}$ in set $\mathcal{S}$, that is, $v_i(\mathcal{S}) = \sum_{j \in \mathcal{S}} v_i(\{j\})$.

Additive functions are fairly simple but also useful, and many expansions exist [11, 13].

Submodular Let $v_i(\mathcal{S}_1 \mid \mathcal{S}_2) \coloneqq v_i(\mathcal{S}_1 \cup \mathcal{S}_2) - v_i(\mathcal{S}_2)$ denote the marginal valuation of agent i for a set $\mathcal{S}_1 \subset \mathcal{G}$ of items over a disjoint set $\mathcal{S}_2 \subset \mathcal{G}$. This valuation function satisfies the submodularity constraint $v_i(\{j\} \mid \mathcal{S}_1 \cup \mathcal{S}_2) \leq v_i(\{j\} \mid \mathcal{S}_1)$ for all agents $i \in \mathcal{A}$, items $j \in \mathcal{G}$ and sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{G}$ of items.

Submodular valuation functions, which encompass the additive ones, have the property that the gain from assigning new items decreases with increasing bundle size. Diminishing returns are a common phenomenon in economics, making submodular functions worthwhile to study [17]. Their relations to matroids [12, 20, 21] make them interesting from a theoretical point of view, too.

In a slight abuse of notation, we sometimes omit curly braces delimiting a set, so we write $v_i(j_1, j_2, \ldots)$ instead of $v_i(\{j_1, j_2, \ldots\})$ for example.

In order to measure and maximise the overall satisfaction of all agents, one needs to combine their valuations. Several options arise here; common choices are the utilitarian social welfare, that is the sum of all valuations [2, 7, 9, 13, 17], and the egalitarian social welfare, that is the minimum of all valuations [7, 13]. We consider a third one, the Nash social welfare.

Problem 2. Given a set $\mathcal{G}$ of items and a set $\mathcal{A}$ of agents with valuation functions $v_i : \mathcal{P}(\mathcal{G}) \to \mathbb{R}_0^+$ and weights $\eta_i \in \mathbb{R}^+$ for all agents $i \in \mathcal{A}$, the *Nash social welfare problem (NSW)* is to find an allocation x^* which maximises the weighted geometric mean of valuations, that is,

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

where $X_{\mathcal{A}}(\mathcal{G})$ is the set of all possible allocations. The problem is called *symmetric* if all agent weights η_i are equal, and *asymmetric* otherwise.

For the techniques employed in later sections, it is beneficial to consider the logarithmic Nash social welfare, that is,

$$\log \mathrm{NSW}(x) = \frac{1}{\sum_{i \in \mathcal{A}} \eta_i} \cdot \sum_{i \in \mathcal{A}} \eta_i \log v_i(x_i)\,, \tag{1}$$

which is a sum instead of a product. The Nash social welfare strike as middle ground between the utilitarian and egalitarian social welfare, which focus on efficiency (height of overall satisfaction) and fairness (how agents value other agents' bundles), respectively. In addition, it exhibits scale-freeness, that is invariance to the scales in which the valuations are expressed. Even though the NSW is $\mathcal{APX}$-hard, approximate solutions largely keep the properties of optimal allocations [3, 6, 8, 18, see also Remark 8].

1.3 Related Work and Contribution

The research on the NSW is rather young and less advanced than the research on other allocation problems. As reference[1], for the submodular utilitarian social welfare problem, a hardness of $\frac{e}{e-1}$ was proven in 2007 [17], and an $\frac{e}{e-1}$-approximation algorithm was shown in 2008 [21] — the additive case is trivially solvable through repeated maximum matchings anyway. For the egalitarian social welfare problem, a randomised $(320\sqrt{n}\log^3 n)$-approximation algorithm for additive valuations [1] and an $(2n-1)$-approximation algorithm for submodular valuations [16] have been devised in 2010 and 2007, respectively. These may not be the best algorithms, though, as the best known lower bound on the approximation factor is 2 [5].

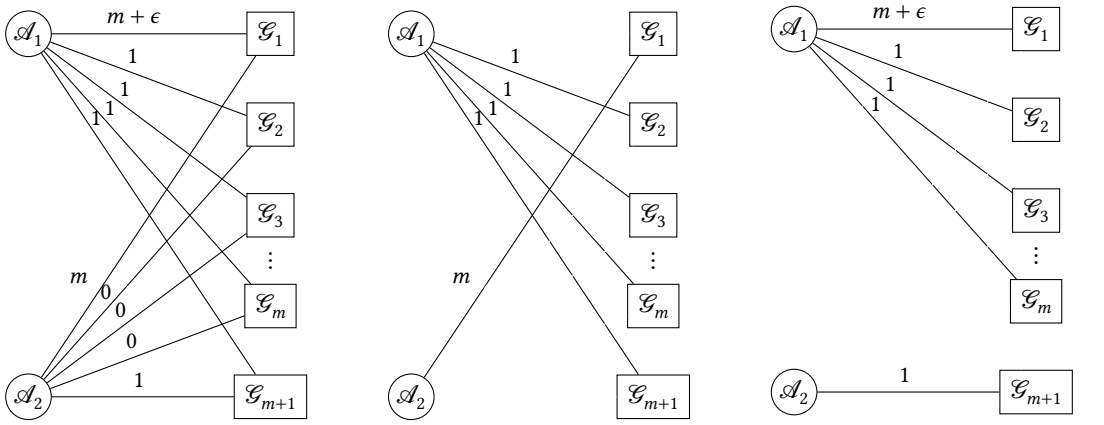
In contrast, for the symmetric additive NSW, an approximation hardness of 1.069 was shown in 2019 [11], but only an 1.45-approximation algorithm has been found in 2018 [3]. For the symmetric submodular NSW, a $(m-n+1)$-approximation algorithm has been devised in 2014 [18]. However, an approximation factor dependant on the number of items is not desirable as the number of items vastly exceeds the number of agents in many applications. Moreover, both approaches exploit the symmetry of the studied problem and fail to extend to the asymmetric case.

Garg, Kulkarni and Kulkarni [13] further the research by contributing two polynomial-time algorithms for the asymmetric NSW. The first one, *SMatch*, computes a $2n$-approximative allocation when the valuation functions are additive. It does so by *s*martly *match*ing agents and items which make up the parts of a bipartite graph. The second one, *RepReMatch*, computes a $2n(\log_2 n + 3)$-approximative allocation when the valuation functions are submodular. It does so by *rep*eatedly computing *match*ings, which then get partly annulled, so that items can be *re*matched.

1.4 Structure of the Paper

We present and analyse SMatch in Section 2 and RepReMatch in Section 3. In Section 4, we give an analysis on the hardness of the submodular NSW. Section 5 contains the summary, an overview of newly published work since 2020, and an outlook on open questions.

[1] The overview is given on the state of research as it was roughly at the end of the year 2019, when Garg, Kulkarni and Kulkarni wrote their paper [13] on which this seminar report is based.



(a) problem instance **(b)** $\mathrm{NSW}(\boldsymbol{x}^*) = \sqrt{m \cdot m}$ **(c)** $\mathrm{NSW}(\boldsymbol{x}) \leq \sqrt{(2m+\epsilon-1)\cdot 1}$

Figure 1: Illustration of Example 3 with $m+1$ items and two equally weighted agents, their valuations shown as edge weights. It demonstrates that naïvely repeated matching without consideration of the future leads to an allocation $\boldsymbol{x}$ which is at best a $\sqrt{m/2}$-approximation of the optimum $\boldsymbol{x}^*$, meaning the approximation factor depends on the number of items.

2 SMatch

2.1 Presentation of the Algorithm

In the case of an equal number of agents and items, i. e., $n = m$, the *additive* NSW can be solved exactly by finding a maximum matching on a bipartite graph with the sets of agents and of items as its parts; the weight of an edge between agent i and item j would be the weighted logarithmic valuation of item j by agent i, i. e. $\eta_i \log v_i(j)$. Should there be more items than agents, then it would seem natural to just repeat this process until all items are assigned. The flaw of this idea is that such a greedy algorithm considers only the valuations of items in the current graph and perhaps the valuations of items already assigned. As the following examples demonstrates, this leads to an algorithm with an approximation factor dependent on the number m of items.

Example 3. There are $m+1$ items $\mathscr{G}_1, \ldots, \mathscr{G}_{m+1}$ and two equally weighted agents $\mathscr{A}_1$ and $\mathscr{A}_2$. Agent $\mathscr{A}_1$ values item $\mathscr{G}_1$ at $m+\epsilon$ and all other items at 1. Agent $\mathscr{A}_2$ values item $\mathscr{G}_1$ at m, item $\mathscr{G}_{m+1}$ at 1 and all other items at 0 (Fig. 1a). In an optimal allocation $\boldsymbol{x}^*$, item $\mathscr{G}_1$ would be assigned to agent $\mathscr{A}_2$ and all other items to agent $\mathscr{A}_1$, resulting in a Nash social welfare of $\sqrt{m \cdot m} = m$ (Fig. 1b). A repeated maximum matching algorithm would greedily assign item $\mathscr{G}_1$ to agent $\mathscr{A}_1$ and item $\mathscr{G}_{m+1}$ to agent $\mathscr{A}_2$ first. Even if all remaining items were going to be assigned to agent $\mathscr{A}_1$, the Nash social welfare would never surpass $\sqrt{(2m+\epsilon-1)\cdot 1} < \sqrt{2m}$ (Fig. 1c). The approximation factor of the output $\boldsymbol{x}$ therefore is at least $\sqrt{m/2}$ and, thus, depends on the number of items.

The geometric mean is high when bundles are valued similarly, wherefore it may be beneficial to give an item to agents who cannot expect many more valuable items in the future instead of to agents who value the item a bit more but do so for other items as well. This is known as the Pigou-Dalton transfer principle [4].

The algorithm SMatch (Algorithm 1) eliminates the flaw by first gaining foresight of the valuations of items assigned after the first matching, thereby achieving an approximation factor of $2n$ (Theorem 7). For a fixed agent i, order the items in descending order of valuations and denote the jth most liked item by $\mathscr{G}_i^j$. We assume a well-defined order, so items of equal valuation shall be ordered further by ID or something similar. SMatch does in fact repeatedly match items. During the first matching, however, the edge weights are defined as $\eta_i \log\big(v_i(j) + u_i/n\big)$ for an edge

Algorithm 1: SMatch for the asymmetric additive NSW

Input : a set $\mathcal{G}$ of m items, a set $\mathcal{A}$ of n agents, additive valuation functions $v_i : \mathcal{P}(\mathcal{G}) \to \mathbb{R}_0^+$ and weights $\eta_i \in \mathbb{R}^+$ for all agents $i \in \mathcal{A}$

Output: $2n$-approximation $\boldsymbol{x} = (\boldsymbol{x}_i)_{i\in\mathcal{A}}$ of an optimal allocation

$u_i \leftarrow v_i(\mathcal{G}_i^{2n+1}, \dots, \mathcal{G}_i^m) \quad \forall i \in \mathcal{A}$ ▷ *estimation of future valuations*

$E \leftarrow \left\{ \left(i, j, \eta_i \log(v_i(j) + u_i/n)\right) \mid i \in \mathcal{A}, j \in \mathcal{G} \right\}$ ▷ *weighted edges using estimation*

$G \leftarrow (\mathcal{A}, \mathcal{G}, E)$ ▷ *bipartite graph*

$\mathcal{M} \leftarrow$ max_weight_matching(G)

$\boldsymbol{x}_i \leftarrow \{ j \mid (i,j) \in \mathcal{M} \} \quad \forall i \in \mathcal{A}$ ▷ *assign according to matching*

$\mathcal{G}^{\text{rem}} \leftarrow \mathcal{G} \setminus \{ j \mid (i,j) \in \mathcal{M} \}$ ▷ *remove assigned items*

while $\mathcal{G}^{\text{rem}} \neq \emptyset$ **do**

  $E \leftarrow \left\{ \left(i, j, \eta_i \log(v_i(j) + v_i(\boldsymbol{x}_i))\right) \mid i \in \mathcal{A}, j \in \mathcal{G}^{\text{rem}} \right\}$ ▷ *weighted edges using only valuations*

  $G \leftarrow (\mathcal{A}, \mathcal{G}^{\text{rem}}, E)$

  $\mathcal{M} \leftarrow$ max_weight_matching(G)

  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i \cup \{ j \mid (i,j) \in \mathcal{M} \} \quad \forall i \in \mathcal{A}$

  $\mathcal{G}^{\text{rem}} \leftarrow \mathcal{G}^{\text{rem}} \setminus \{ j \mid (i,j) \in \mathcal{M} \}$

end while

return $\boldsymbol{x}$

between agent i and item j. The addend u_i serves as estimation of the valuation of items assigned after the first matching (Lemma 5) and is defined as

$$u_i := \min_{\substack{\mathcal{S}\subset\mathcal{G} \\ |\mathcal{S}|\leq 2n}} \{v_i(\mathcal{G} \setminus \mathcal{S})\} = v_i(\mathcal{G}_i^{2n+1}, \dots, \mathcal{G}_i^m). \tag{2}$$

Note that the set $\mathcal{S}$ realizing the minimum of $v_i(\mathcal{G} \setminus \mathcal{S})$ may have less than $2n$ elements if there are less than $2n$ items in total or if some items have a valuation of zero. From the second matching onwards, the edge weights are defined as $\eta_i \log\big(v_i(j) + v_i(\boldsymbol{x}_i)\big)$, where $\boldsymbol{x}_i$ is the continuously updated bundle of agent i. The addend $v_i(\boldsymbol{x}_i)$ could lead to better allocations in applications but does not improve the approximation factor asymptotically and so is of no further interest in the analysis.

2.2 Analysis of the Algorithm

To calculate the approximation factor of SMatch, we first need to establish a lower bound on the valuation of single items. For convenience, we order the items of any set $\mathcal{S} = \{j^1, j^2, \dots\}$ in descreasing order of valuation, so that it holds $v_i(j^k) \geq v_i(j^{k'})$ for all $k < k'$. Note that if set $\mathcal{S}$ happens to be a bundle, then item j^k was assigned according to the kth matching.

Lemma 4. *For each agent $i \in \mathcal{A}$ and her final bundle $\boldsymbol{x}_i = \{h_i^1, \dots, h_i^{\tau_i}\}$, it holds $v_i(h_i^t) \geq v_i(\mathcal{G}_i^{tn})$ for all $t = 1, \dots, \tau_i$.*

Proof. Before the tth matching, no more than $(t-1)n$ items out of the tn most highly valued items $\mathcal{G}_i^1, \dots, \mathcal{G}_i^{tn}$ have been assigned in previous matchings since at most n many out of those items are assigned each time. Because of the tth matching, at most $n-1$ more could be assigned to other agents, leaving at least one item of $\mathcal{G}_i^1, \dots, \mathcal{G}_i^{tn}$ unassigned. Since $v_i(\mathcal{G}_i^k) \geq v_i(\mathcal{G}_i^{tn})$ for all $k \leq tn$ by definition of $\mathcal{G}_i^k$, the lemma follows. □

We can now establish $u_i/n = v_i(\mathcal{G}_i^{2n+1}, \dots, \mathcal{G}_i^m)/n$ as lower bound on the valuations of items assigned after the first matching.

Lemma 5. *For each agent $i \in \mathcal{A}$ and her final bundle $\boldsymbol{x}_i = \{h_i^1, \dots, h_i^{\tau_i}\}$, it holds $v_i(h_i^2, \dots, h_i^{\tau_i}) \geq u_i/n$.*

Proof. By Lemma 4 and definition of $\mathcal{G}_i^{tn}$, every item h_i^t is worth at least as much as each item $\mathcal{G}_i^{tn+k}$ with $k \in \{1, \dots, n\}$ and, consequently, its valuation $v_i(h_i^t)$ is at least as high as the mean valuation $\frac{1}{n} v_i(\mathcal{G}_i^{tn+1}, \dots, \mathcal{G}_i^{tn+n})$. Also, it holds $\tau_i n + n \geq m$ since agent i gets assigned $\tau_i \geq \left\lfloor \frac{m}{n} \right\rfloor \geq \frac{m}{n} - 1$ many items. This yields

$$v_i(h_i^2, \dots, h_i^{\tau_i}) = \sum_{t=2}^{\tau_i} v_i(h_i^t) \geq \sum_{t=2}^{\tau_i} \frac{1}{n} v_i(\mathcal{G}_i^{tn+1}, \dots, \mathcal{G}_i^{tn+n}) \geq \frac{1}{n} v_i(\mathcal{G}_i^{2n+1}, \dots, \mathcal{G}_i^{m}) = \frac{u_i}{n}. \tag{3}$$

□

Remark 6. In Lemma 5, we assumed non-zero valuations for all items, hence the bundle lengths of $\tau_i \geq \left\lfloor \frac{m}{n} \right\rfloor$. Of course, one would not assign items to agents who value them at zero in an actual program. Nevertheless, Lemma 5 still holds inasmuch as additional zeroes do not change the sum in Eq. (3).

This allows us to calculate an approximation factor for SMatch by comparing its output with an optimal allocation $\boldsymbol{x}^*$.

Theorem 7. *SMatch has an approximation factor of* $2n$.

Proof. Lemma 5 can be plugged into the logarithmic Nash social welfare:

$$\log \mathrm{NSW}(\boldsymbol{x}) = \frac{1}{\sum_{i \in \mathcal{A}} \eta_i} \cdot \sum_{i \in \mathcal{A}} \eta_i \log v_i(h_i^1, \dots, h_i^{\tau_i}) \tag{4}$$

$$= \frac{1}{\sum_{i \in \mathcal{A}} \eta_i} \cdot \sum_{i \in \mathcal{A}} \eta_i \log\big(v_i(h_i^1) + v_i(h_i^2, \dots, h_i^{\tau_i})\big) \tag{5}$$

$$\geq \frac{1}{\sum_{i \in \mathcal{A}} \eta_i} \cdot \sum_{i \in \mathcal{A}} \eta_i \log\big(v_i(h_i^1) + u_i/n\big) \tag{6}$$

Notice that the first matching of SMatch maximises the sum in Eq. (6). Thus, assigning all agents i their respective most highly valued item g_i^1 from an optimal bundle $\boldsymbol{x}_i^* = \{g_i^1, \dots, g_i^{\tau_i^*}\}$ yields the even lower bound

$$\log \mathrm{NSW}(\boldsymbol{x}) \geq \frac{1}{\sum_{i \in \mathcal{A}} \eta_i} \cdot \sum_{i \in \mathcal{A}} \eta_i \log\big(v_i(g_i^1) + u_i/n\big). \tag{7}$$

Recall the definition of u_i from Eq. (2). Consider a slightly modified variant:

$$u_i = v_i(\mathcal{G} \setminus \mathcal{S}_i) \text{ with } \mathcal{S}_i \coloneqq \operatorname*{arg\,min}_{\substack{\mathcal{S} \subset \mathcal{G} \\ |\mathcal{S}| \leq 2n}} \{v_i(\mathcal{G} \setminus \mathcal{S})\} \tag{8}$$

Now consider the set $\mathcal{S}_i^*$ of the (at most) $2n$ most highly valued items in the optimal bundle $\boldsymbol{x}_i^*$, i. e.

$$\mathcal{S}_i^* \coloneqq \operatorname*{arg\,min}_{\substack{\mathcal{S} \subset \boldsymbol{x}_i^* \\ |\mathcal{S}| \leq 2n}} \{v_i(\boldsymbol{x}_i^* \setminus \mathcal{S})\}. \tag{9}$$

It holds $v_i(g_i^1) \geq \frac{1}{2n} v_i(\mathcal{S}_i^*)$ because of $v_i(g_i^1) \geq v_i(j)$ for all items $j \in \mathcal{S}_i^*$. Furthermore, it holds $u_i = v_i(\mathcal{G} \setminus \mathcal{S}_i) \geq v_i(\boldsymbol{x}_i^* \setminus \mathcal{S}_i) \geq v_i(\boldsymbol{x}_i^* \setminus \mathcal{S}_i^*)$. We can insert these two inequalities into Eq. (7) and prove the theorem thereby:

$$\log \mathrm{NSW}(\boldsymbol{x}) \geq \frac{1}{\sum_{i \in \mathcal{A}} \eta_i} \cdot \sum_{i \in \mathcal{A}} \eta_i \log\left(\frac{v_i(\mathcal{S}_i^*)}{2n} + \frac{v_i(\boldsymbol{x}_i^* \setminus \mathcal{S}_i^*)}{n}\right) \tag{10}$$

$$\geq \frac{1}{\sum_{i \in \mathcal{A}} \eta_i} \cdot \sum_{i \in \mathcal{A}} \eta_i \log\left(\frac{v_i(\boldsymbol{x}_i^*)}{2n}\right) \tag{11}$$

$$= \log\left(\frac{\mathrm{NSW}(\boldsymbol{x}^*)}{2n}\right) \tag{12}$$

□

The analysis is asymptotically tight. It is possible to design an instance for the asymmetric additive NSW such that SMatch achieves an approximation ratio approaching $n/2$. It remains to be shown whether the symmetric additive NSW leads to an asymptotically equally bad approximation factor. [13, Section 6.3]

Remark 8. SMatch produces *fair* allocations which are *envy-free up to one item (EF1).* An allocation $\boldsymbol{x}$ is EF1 if, for every pair $(i, i') \in \mathscr{A}^2$ of agents, one needs to remove at most one item from the bundle $\boldsymbol{x}_{i'}$ of agent i' so that agent i does not want to swap bundles. In other words, either it holds $v_i(\boldsymbol{x}_i) \geq v_{i'}(\boldsymbol{x}_{i'})$ or there is an item $j \in \boldsymbol{x}_{i'}$ such that $v_i(\boldsymbol{x}_i) \geq v_{i'}(\boldsymbol{x}_{i'} \setminus \{j\})$. [13, Section 5.2]

However, SMatch does not produce allocations which are *Pareto-optimal (PO)*, another popular fairness property. An allocation $\boldsymbol{x}$ is PO if there is no other allocation $\boldsymbol{x}'$ where every agent is at least as well off and one agent is strictly better off, i. e., it does not hold $v_i(\boldsymbol{x}'_i) \geq v_i(\boldsymbol{x}_i)$ for all agents $i \in \mathscr{A}$ and $v_{i'}(\boldsymbol{x}'_{i'}) > v_{i'}(\boldsymbol{x}_{i'})$ for some agent $i' \in \mathscr{A}$. [13, Remark 5.2]

3 RepReMatch

3.1 Presentation of the Algorithm

The algorithm SMatch estimates the valuation of the lowest-value items by determining the set of highest-value items and then valuing the remaining items. Unfortunately, this approach does not work for general *submodular* valuation functions because taking the set of highest-value items away does not necessarily leave a set of lowest-value items. In fact, it can be shown [20] that determining the set of lowest-value items is approximable only within a factor of $\Omega(\sqrt{m/\ln m})$.

For this reason, the algorithm RepReMatch (Algorithm 2) relies on an approach with three phases, thereby achieving an approximation factor of $2n(\log_2 n + 3)$ (Theorem 16). In phase I, a sufficiently big set of high-value items is determined through repeated matchings. This phase serves merley to determine this set, so items are assigned temporarily only. The edge weights reflect this by taking the valuations of just single items into account.

Algorithm 2: RepReMatch for the asymmetric submodular NSW

Input : a set $\mathcal{G}$ of m items, a set $\mathcal{A}$ of n agents, submodular valuation functions $v_i : \mathcal{P}(\mathcal{G}) \to \mathbb{R}_0^+$ and weights $\eta_i \in \mathbb{R}^+$ for all agents $i \in \mathcal{A}$

Output: $2n(\log_2 n + 3)$-approximation $\boldsymbol{x}^{\mathrm{III}} = (\boldsymbol{x}_i^{\mathrm{III}})_{i\in\mathcal{A}}$ of an optimal allocation

Phase I:

$\boldsymbol{x}_i^{\mathrm{I}} \leftarrow \emptyset \quad \forall i \in \mathcal{A}$ ▷ *initialise temporary allocation*

$\mathcal{G}^{\mathrm{rem}} \leftarrow \mathcal{G}$ ▷ *set of unassigned items*

for $t \leftarrow 1, \dots, \lceil \log_2 n \rceil + 1$ **do**

  if $\mathcal{G}^{\mathrm{rem}} \neq \emptyset$ **then**

    $E \leftarrow \left\{ \left(i, j, \eta_i \log v_i(j)\right) \mid i \in \mathcal{A}, j \in \mathcal{G}^{\mathrm{rem}} \right\}$ ▷ *weighted edges using val. of single item*

    $G \leftarrow (\mathcal{A}, \mathcal{G}^{\mathrm{rem}}, E)$ ▷ *bipartite graph*

    $\mathcal{M} \leftarrow \mathrm{max\_weight\_matching}(G)$

    $\boldsymbol{x}_i^{\mathrm{I}} \leftarrow \boldsymbol{x}_i^{\mathrm{I}} \cup \{j\} \quad \forall (i,j) \in \mathcal{M}$ ▷ *assign temporarily according to matching*

    $\mathcal{G}^{\mathrm{rem}} \leftarrow \mathcal{G}^{\mathrm{rem}} \setminus \{ j \mid (i,j) \in \mathcal{M} \}$ ▷ *remove assigned items*

  end if

end for

Phase II:

$\boldsymbol{x}_i^{\mathrm{II}} \leftarrow \emptyset \quad \forall i \in \mathcal{A}$ ▷ *put allocation $\boldsymbol{x}^{\mathrm{I}}$ away & initialise new, definite one*

while $\mathcal{G}^{\mathrm{rem}} \neq \emptyset$ **do**

  $E \leftarrow \left\{ \left(i, j, \eta_i \log v_i(\boldsymbol{x}_i^{\mathrm{II}} \cup \{j\})\right) \mid i \in \mathcal{A}, j \in \mathcal{G}^{\mathrm{rem}} \right\}$ ▷ *valuation of item & current bundle*

  $G \leftarrow (\mathcal{A}, \mathcal{G}^{\mathrm{rem}}, E)$

  $\mathcal{M} \leftarrow \mathrm{max\_weight\_matching}(G)$

  $\boldsymbol{x}_i^{\mathrm{II}} \leftarrow \boldsymbol{x}_i^{\mathrm{II}} \cup \{j\} \quad \forall (i,j) \in \mathcal{M}$ ▷ *assign definitely according to matching*

  $\mathcal{G}^{\mathrm{rem}} \leftarrow \mathcal{G}^{\mathrm{rem}} \setminus \{ j \mid (i,j) \in \mathcal{M} \}$

end while

Phase III:

$\mathcal{G}^{\mathrm{rem}} \leftarrow \bigcup_{i\in\mathcal{A}} \boldsymbol{x}_i^{\mathrm{I}}$ ▷ *release items assigned in phase I*

$E \leftarrow \left\{ \left(i, j, \eta_i \log v_i(\boldsymbol{x}_i^{\mathrm{II}} \cup \{j\})\right) \mid i \in \mathcal{A}, j \in \mathcal{G}^{\mathrm{rem}} \right\}$ ▷ *valuation of item & current bundle*

$G \leftarrow (\mathcal{A}, \mathcal{G}^{\mathrm{rem}}, E)$

$\mathcal{M} \leftarrow \mathrm{max\_weight\_matching}(G)$

$\boldsymbol{x}_i^{\mathrm{III}} \leftarrow \boldsymbol{x}_i^{\mathrm{II}} \cup \{j\} \quad \forall (i,j) \in \mathcal{M}$ ▷ *initialise final allocation & assign def. according to matching*

$\mathcal{G}^{\mathrm{rem}} \leftarrow \mathcal{G}^{\mathrm{rem}} \setminus \{ j \mid (i,j) \in \mathcal{M} \}$

$\boldsymbol{x}^{\mathrm{III}} \leftarrow \mathrm{arbitrary\_allocation}(\mathcal{A}, \mathcal{G}^{\mathrm{rem}}, \boldsymbol{x}^{\mathrm{III}}, (v_i)_{i\in\mathcal{A}})$ ▷ *assign remainder of items arbitrarily*

return $\boldsymbol{x}^{\mathrm{III}}$

In phase II, the remaining items are assigned definitely through repeated matchings. Consequently, each edge weight is updated in each round to reflect the valuation of both the respective item and the items assigned so far.

In phase III, the high-value items assigned in phase I are released. One maximum weight matching is calculated, and the matched items are assigned accordingly. Similarly to the previous phase, each edge weight uses the valuation of both the respective item and the items assigned so far, i. e., the bundle from phase II. The remaining released items are assigned arbitrarily.

3.2 Analysis of the Algorithm

We use the term *round* to refer to the iterations of the loops in the phases I and II. For ease of notation, we refer to the moment right before the first iteration in phase II as round 0. We start by analysing phase II as it is the first phase with definitive assignments. To this end, we introduce two types of item sets.

Definition 9. Let x_i^* be an optimal bundle of an agent $i \in \mathscr{A}$. For any round $r \geq 1$ in phase II, the set $\mathscr{L}_{i,r} \subset x_i^*$ of *lost items* is the set of all items $j \in x_i^*$ assigned to other agents $i' \neq i$ in that round.

Definition 10. Let x_i^* be an optimal bundle of an agent $i \in \mathscr{A}$ and $x_i^{\mathrm{II}} = \{h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}\}$ be her bundle at the end of phase II. The set of *optimal and attainable items* is defined as $\overline{x}_{i,0}^* := x_i^* \setminus \bigcup_{i' \in \mathscr{A}} x_{i'}^{\mathrm{I}}$ for round 0 and as $\overline{x}_{i,r}^* := \overline{x}_{i,r-1}^* \setminus (\mathscr{L}_{i,r} \cup \{h_i^{r-1}\})$ for round $r = 1, \dots, \tau_i^{\mathrm{II}}$.

We denote their sizes by $\ell_{i,r} := |\mathscr{L}_{i,r}|$ and $\overline{\tau}_{i,r}^* := |\overline{x}_{i,r}^*|$, respectively. First, we give a lower bound on the valuations of optimal and attainable items.

Lemma 11. *For each agent $i \in \mathscr{A}$ and her bundle $x_i^{\mathrm{II}} = \{h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}\}$, it holds in all rounds $r = 2, \dots, \tau_i^{\mathrm{II}}$ of phase II that*

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

Proof. The proof by Garg, Kulkarni and Kulkarni [13, pp. 13 sq.] consists of a lengthy induction rich in formulae and case differentiations. This is why we give a different but intuitive approach.

Writing the left-hand side of the lemma out in full gives

$$v_i(\overline{x}_{i,r}^* \mid h_i^1, \dots, h_i^{r-1}) = v_i(\overline{x}_{i,r}^* \cup \{h_i^1, \dots, h_i^{r-1}\}) - v_i(h_i^1, \dots, h_i^{r-1}) , \tag{13}$$

which explains the second subtrahend on the right-hand side.

Next, we show a lower bound on $v_i(\overline{x}_{i,r}^* \cup \{h_i^1, \dots, h_i^{r-1}\})$. The items which are optimal and attainable in round r were so in round $r-1$, too. Additionally, the optimal but lost items of round r were attainable in round $r-1$ as well. The item h_i^r assigned to agent i in round r was also attainable in round $r-1$ and may also be optimal. Therefore, it holds $(\overline{x}_{i,r}^* \cup \{h_i^{r-1}\}) \supset (\overline{x}_{i,r-1}^* \setminus \mathscr{L}_{i,r})$ and

$$v_i(\overline{x}_{i,r}^* \cup \{h_i^1, \dots, h_i^{r-1}\}) \geq v_i(\overline{x}_{i,r-1}^* \setminus \mathscr{L}_{i,r} \cup \{h_i^1, \dots, h_i^{r-2}\}) \geq v_i(\overline{x}_{i,r-1}^* \cup \{h_i^1, \dots, h_i^{r-2}\}) - v_i(\mathscr{L}_{i,r}) . \tag{14}$$

The second inequality makes intuitively sense when one thinks of diminishing returns: The gain from adding the set $\mathscr{L}_{i,r}$ to the items in $\overline{x}_{i,r-1}^* \cup \{h_i^1, \dots, h_i^{r-1}\}$ cannot be higher than its subtracted valuation over the empty set. If we now recursively apply Eq. (14), we eventually arrive at

$$v_i(\overline{x}_{i,r}^* \cup \{h_i^1, \dots, h_i^{r-1}\}) \geq v_i(\overline{x}_{i,1}^*) - \sum_{r'=1}^{r-1} v_i(\mathscr{L}_{i,r'+1}) . \tag{15}$$

Lastly, we give an upper bound on $v_i(\mathscr{L}_{i,r'+1})$. The valuation $v_i(j \mid h_i^1, \dots, h_i^{r'-1})$ of any item $j \in \mathscr{L}_{i,r'+1}$ in round r' cannot have been higher than $v_i(h_i^{r'} \mid h_i^1, \dots, h_i^{r'-1})$ as otherwise item $h_i^{r'}$ would not have been assigned before item j. From this follows

$$v_i(\mathscr{L}_{i,r'+1}) \leq \ell_{i,r'+1} \cdot v_i(h_i^{r'} \mid h_i^1, \dots, h_i^{r'-1}) \,. \tag{16}$$

Combining Eqs. (13), (15) and (16) proves the lemma. □

The lemma can be used to find a lower bound on the marginal valuation of the items actually assigned in each round r.

Corollary 12. *From Lemma 11 follows*

$$v_i(h_i^r \mid h_i^1, \dots, h_i^{r-1}) \geq \left(v_i(\overline{x}_{i,1}^*) - \sum_{r'=1}^{r-1} \ell_{i,r'+1} \cdot v_i(h_i^{r'} \mid h_i^1, \dots, h_i^{r'-1}) - v_i(h_i^1, \dots, h_i^{r-1})\right) \Big/ \overline{\tau}_{i,r}^* \,.$$

Proof. Remember that valuation functions are monotonic if $v_i(\mathcal{S}_1) \leq v_i(\mathcal{S}_2)$ holds for all sets $\mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{G}$. Induction shows that there must be an item $j \in \mathcal{S}_2$ with valuation $v_i(j) \geq v_i(\mathcal{S}_2)/|\mathcal{S}_2|$, otherwise it would hold $v_i(\emptyset) > 0$. Applied to Lemma 11, this means that there must be an item $j \in \overline{x}_{i,r}^*$ with a marginal valuation of at least $v_i(\overline{x}_{i,r}^* \mid h_i^1, \dots, h_i^{r-1})/\overline{\tau}_{i,r}^*$. As item h_i^r was the one to be assigned, its marginal valuation cannot be smaller. □

This, finally, enables us to give a lower bound on the valuation of the bundles x_i^{II}.

Lemma 13. *For each agent $i \in \mathscr{A}$ and her bundle $x_i^{\text{II}} = \{h_i^1, \dots, h_i^{\tau_i^{\text{II}}}\}$, it holds*

$$v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}}) \geq v_i(\overline{x}_{i,1}^*)/n \,.$$

Proof. In each round $r = 1, \dots, \tau_i^{\text{II}}$, $\ell_{i,r}$ optimal and attainable items of agent i are assigned to other agents. As there are n agents in total, $n-1$ is an upper bound on $\ell_{i,r}$. Furthermore, after τ_i^{II} rounds, the number $\overline{\tau}_{i,\tau_i^{\text{II}}}^*$ of optimal and attainable items is at most $n-1 \leq n$ elsewise agent i would have been assigned yet another item. Together with Corollary 12, this proves the lemma:

$$\begin{aligned}
v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}}) &= v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1}) + v_i(h_i^{\tau_i^{\text{II}}} \mid h_i^1, \dots, h_i^{\tau_i^{\text{II}}}) && (17)\\
&\geq v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1}) + \Bigg(v_i(\overline{x}_{i,1}^*) - \sum_{r'=1}^{\tau_i^{\text{II}}-1} \ell_{i,r'+1} \cdot v_i(h_i^{r'} \mid h_i^1, \dots, h_i^{r'-1}) \\
&\qquad\qquad - v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1})\Bigg) \Big/ \overline{\tau}_{i,\tau_i^{\text{II}}}^* && (18)\\
&\geq v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1}) + \Bigg(v_i(\overline{x}_{i,1}^*) - \sum_{r'=1}^{\tau_i^{\text{II}}-1} (n-1) v_i(h_i^{r'} \mid h_i^1, \dots, h_i^{r'-1}) \\
&\qquad\qquad - v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1})\Bigg) \Big/ n && (19)\\
&\geq v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1}) + \left(v_i(\overline{x}_{i,1}^*) - (n-1) v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1}) - v_i(h_i^1, \dots, h_i^{\tau_i^{\text{II}}-1})\right) \Big/ n && (20)\\
&= v_i(\overline{x}_{i,1}^*)/n && (21)
\end{aligned}$$

□

After having obtained a lower bound on the valuation of items assigned in phase II, we need a lower bound for phase III as well. Therefor we introduce a third type of item set.

Definition 14. Let $x_i^* = \{g_i^1, \dots, h_i^{\tau_i^*}\}$ be an optimal bundle of some agent $i \in \mathscr{A}$. The set of *overly good* items is defined as $\mathscr{G}_i^+ := \{\, j \in \mathscr{G} \mid v_i(j) \geq v_i(g_i^1) \,\}$.

Lemma 15. *In phase III, there exists a matching such that each agent $i \in \mathscr{A}$ is matched to one of her overly good items in the set $\bigcup_{i' \in \mathscr{A}} \boldsymbol{x}_{i'}^{\mathrm{I}}$ of released items.*

Proof. If all items were matched in phase I, i. e., $\bigcup_{i' \in \mathscr{A}} \boldsymbol{x}_{i'}^{\mathrm{I}} = \mathscr{G}$, then all optimal items are released in phase III and each agent can be matched to one; the lemma is proven immediately. If not, imagine for some t that only the items assigned in the first t rounds of phase I were released. Now choose some matching $\mathscr{M}_t$ with the following properties:

1. If for an agent i all overly good items were amongst the released items, she gets matched with an overly good item $j \in \mathscr{G}_i^+$.
2. The number of agents matched with one of their overly good items is maximal amongst all matchings fulfilling property 1.

Property 1 is always satisfiable as the union of k many sets $\mathscr{G}_i^+$ contains k different items g_i^1, which can be matched with agents i. Property 2 leads to all agents being matched with an overly good item when $t = \lceil \log_2 n \rceil + 1$, i. e. the number of rounds in phase I, whence the lemma follows. To prove this, we denote by $\mathscr{A}_t^-$ the set of agents who are *not* matched with one of their overly good items, and show by induction on t that it holds $|\mathscr{A}_t^-| \leq n/2^t$.

In the base case $t = 1$, none of the items are assigned initially. Denote by α the number of agents who were not assigned an overly good item in the first round of phase I. If $\alpha \leq n/2$, then a matching $\mathscr{M}_1$ obviously exists and the base case is immediately proven. Otherwise, all items from at least α many sets $\mathscr{G}_i^+$ got assigned to someone. Again: Each set $\mathscr{G}_i^+$ contains the item g_i^1, so the union of these sets contains at least α items which can be matched with at least α agents upon release. This then leaves at most $n - \alpha < n/2$ agents not matched with an overly good item.

For the induction hypothesis, we assume that the statement holds true for all rounds up to some t. In the induction step $t \to t+1$, by property 1, there is at least one unassigned, overly good item in each set $\mathscr{G}_{i'}^+$ of all agents $i' \in \mathscr{A}_t^-$ at the start of round $t+1$. Analogously to the base case, for at least half of those agents i', these unassigned items will be assigned to either them or someone else, and it can be argued accordingly. By the induction hypothesis, it holds $|\mathscr{A}_{t+1}^-| \leq |\mathscr{A}_t^-|/2 \leq (n/2^t)/2 = n/2^{t+1}$. □

This allows us to calculate an approximation factor for RepReMatch by comparing its output with an optimal allocation $\boldsymbol{x}^*$.

Theorem 16. *RepReMatch has an approximation factor of $2n(\log_2 n + 3)$.*

Proof. By Lemma 15, we can assign each agent i an overly good item $j_i^+ \in \mathscr{G}_i^+$ in the beginning of phase III. RepReMatch maximises the logarithmic Nash social welfare, so

$$\log \mathrm{NSW}(\boldsymbol{x}^{\mathrm{III}}) \geq \frac{1}{\sum_{i \in \mathscr{A}} \eta_i} \cdot \sum_{i \in \mathscr{A}} \eta_i \log v_i(j_i^+, h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}) \tag{22}$$

is a lower bound on the logarithmic NSW after the first matching in phase III, with $\boldsymbol{x}_i^{\mathrm{II}} = \{h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}\}$ being the bundle of agent i at the end of phase II.

Item j_i^+ was released in phase III, which means it was assigned in phase I, implying $j_i^+ \in \boldsymbol{x}_i^* \setminus \overline{\boldsymbol{x}}_{i,0}^*$ and, subsequently, $j_i^+ \in (\boldsymbol{x}_i^* \setminus \overline{\boldsymbol{x}}_{i,0}^*) \cup \mathscr{L}_{i,1}$. Phase I runs for at most $\lceil \log_2 n \rceil + 1$ rounds, and at most n items are assigned in each iteration. Therefore, at most $n(\log_2 n + 2)$ optimal items are assigned in that phase, i. e., $|\boldsymbol{x}_i^* \setminus \overline{\boldsymbol{x}}_{i,0}^*| \leq n(\log_2 n + 2)$. As in Lemma 13, it also holds $n \geq \ell_{i,1} = |\mathscr{L}_{i,1}|$. Together with the monotonicity of the valuation functions, this yields

$$v_i(j_i^+, h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}) \geq v_i(j_i^+) \geq \frac{v_i\big((\boldsymbol{x}_i^* \setminus \overline{\boldsymbol{x}}_{i,0}^*) \cup \mathscr{L}_{i,1}\big)}{n(\log_2 n + 3)} \tag{23}$$

as lower bound on the valuations of bundles. Moreover, Lemma 5 and the monotonicity of the valuation functions functions yield

$$v_i(j_i^+, h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}) \geq v_i(h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}) \geq \frac{v_i(\bar{x}_{i,1}^*)}{n} \geq \frac{v_i(\bar{x}_{i,1}^*)}{n(\log_2 n + 3)} = \frac{v_i(\bar{x}_{i,0}^* \setminus \mathscr{L}_{i,1})}{n(\log_2 n + 3)} \tag{24}$$

as yet another lower bound. The mean of Eqs. (23) to (24) and the monotonicity of the valuation functions give the concise lower bound

$$\begin{aligned} v_i(j_i^+, h_i^1, \dots, h_i^{\tau_i^{\mathrm{II}}}) &\geq \frac{1}{2}\left(\frac{v_i\big((x_i^* \setminus \bar{x}_{i,0}^*) \cup \mathscr{L}_{i,1}\big)}{n(\log_2 n + 3)} + \frac{v_i\big(\bar{x}_{i,0}^* \setminus \mathscr{L}_{i,1}\big)}{n(\log_2 n + 3)} \right) && (25) \\ &\geq \frac{1}{2} \cdot \frac{v_i\big(((x_i^* \setminus \bar{x}_{i,0}^*) \cup \mathscr{L}_{i,1}) \cup (\bar{x}_{i,0}^* \setminus \mathscr{L}_{i,1})\big)}{n(\log_2 n + 3)} && (26) \\ &= \frac{v_i(x_i^*)}{2n(\log_2 n + 3)} \,. && (27) \end{aligned}$$

We can insert this lower bound into Eq. (22) and prove the theorem thereby:

$$\log \mathrm{NSW}(x^{\mathrm{III}}) \geq \frac{1}{\sum_{i \in \mathscr{A}} \eta_i} \cdot \sum_{i \in \mathscr{A}} \eta_i \log\left(\frac{v_i(x_i^*)}{2n(\log_2 n + 3)} \right) = \log\left(\frac{\mathrm{NSW}(x^*)}{2n(\log_2 n + 3)} \right) \tag{28}$$

□

overly good ⟶ outstanding?

4 Hardness of Approximation

Garg, Kulkarni and Kulkarni [13, Sction 4] provide the following hardness result.

Theorem 17. *The submodular NSW is not approximable within a factor of* $\frac{e}{e-1}$ *in polynomial time unless* $\mathcal{P} = \mathcal{NP}$*, even when the agents have equal weights and valuation functions.*

Proof. Consider the related utilitarian social welfare problem[1].

Problem 18. Given a set $\mathcal{G}$ of items and a set $\mathcal{A}$ of agents with valuation functions $v_i : \mathcal{P}(\mathcal{G}) \to \mathbb{R}_0^+$ for all agents $i \in \mathcal{A}$, the symmetric submodular *utilitarian social welfare problem (USW)* is to find an allocation $\boldsymbol{x}^*$ which maximises the sum of valuations, that is,

$$\boldsymbol{x}^* \overset{!}{=} \arg\max_{\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{A}}(\mathcal{G})} \{\mathrm{USW}(\boldsymbol{x})\} \quad \text{with } \mathrm{USW}(\boldsymbol{x}) \coloneqq \sum_{i \in \mathcal{A}} v_i(\boldsymbol{x}_i)$$

where $\boldsymbol{X}_{\mathcal{A}}(\mathcal{G})$ is the set of all possible allocations of the items in $\mathcal{G}$ amongst n agents.

Note that this problem is identical to the symmetric submodular NSW except for the sum in the objective function instead of a product. We will exploit this fact to calculate the Nash social welfare of instances for USW. Khot et al. [17] supply a polynomial-time reduction of USW from the following problem:

Problem 19. Given a graph $G = (V, E)$ and a constant $c \leq 1$, the *c-Gap-Max-3-Colouring problem* is to decide whether, for any 3-colouring of graph G which maximises the number of edges with different coloured endpoints, the number of such edges is $|E|$ (*Yes instance*) or $c|E|$ and below (*No instance*).

Proposition 20. *There exists a constant* $c \leq 1$ *such that the c-Gap-Max-3-Colouring problem is* $\mathcal{NP}$*-hard.*

Reducing an instance of the c-Gap-Max-3-Colouring problem yields an instance of the symmetric submodular USW with identical valuation functions. Its properties are as follows:

Yes instance The utilitarian social welfare is nC because every agent values her bundle at n, whereby C is a constant depending on the input graph. The Nash social welfare of the instance would be C.

No instance The utilitarian social welfare is $\frac{e-1}{e}nC$. Applying the inequality of arithmetic and geometric means, i. e., $(x_1 + \cdots + x_n)/n \geq \sqrt[n]{x_1 \cdots x_n}$ for all nonnegative numbers $x_1, \ldots, x_n \in \mathbb{R}_0^+$, reveals that the Nash social welafre of the instance is at most $\frac{e-1}{e}C$.

Thereout follows that the submodular NSW, even when symmetric and with identical valuation functions, cannot be approximated within a factor better than $\frac{e}{e-1}$; otherwise one could decide the c-Gap-Max-3-Colouring in polynomial time by checking whether the corresponding NSW instance has a value above $\frac{e-1}{e}C$. □

For a constant number of agents, Garg, Kulkarni and Kulkarni [13, Section 5.1] describe a family $(A_\epsilon)_{\epsilon>0}$ of algorithms for the asymmetric submodular NSW where each algorithm A_ϵ achieves an approximation factor of $\frac{e}{e-1} + \epsilon$.

[1] Garg, Kulkarni and Kulkarni (and many others) call Problem 18 the 'Allocation problem'. We changed the name to match the naming scheme of the NSW problem and to avoid confusion, as both problems are about finding allocations.

5 Conclusion

5.1 Summary

The asymmetric Nash social welfare problem (NSW) asks to find an allocation of unsharable and indivisible items amongst agents such that the weighted geometric mean of their valuations is maximised. In this seminar report, we presented two polynomial-time algorithms for the asymmetric NSW, based on a paper by Garg, Kulkarni and Kulkarni [13]. The novelty lies in both algorithms having an approximation factor dependant on the number n of agents but not on the number m of items.

The first algorithm, SMatch, finds a $2n$-approximative allocation if the valuation functions are additive. It does so by repeatedly matching agents with items within a weighted bipartite graph. For the very first matching, the edge weights incorporate an estimation of the valuation of future items. The output allocation is envy-free up to one item.

The second algorithm, RepReMatch, finds a $2n(\log_2 n + 3)$-approximative allocation if the valuation functions are submodular. In phase I, a set of high-value items is determined through repeated matchings and then put away. In phase II, agents are repeatedly matched with the remaining items. In phase III, the high-value items are finally assigned to the agents.

Lastly, it was shown that any polynomial-time algorithm for the submodular NSW must have an approximation factor of at least $\frac{e}{e-1}$ unless $\mathscr{P} = \mathscr{NP}$. This holds even when the problem is symmetric and the valuation functions equal.

5.2 Recent Work and Open Questions

Garg, Kulkarni and Kulkarni [13] conjecture that SMatch has a better approximation factor for the symmetric additive NSW though neither were they able to prove a better factor nor could they prove the tightness of their given analysis. Additionally, they forbore to prove the tightness of the analysis of RepReMatch, and we suspect there to be quite some constants to be saved — admittedly, the benefits of a more thorough analysis are questionable.

To identify more general open questions, we need to take a look at new publications since 2020, the year of the publication of the paper by Garg, Kulkarni and Kulkarni.

- XOS functions encompass the submodular functions. Unfortunately, RepReMatch does not guarantee an approximation factor independent of the number of items even for the symmetric XOS NSW [13, Section 6.2]. Barman et al. [4] used both RepReMatch and the discrete moving-knife method to get an $\mathcal{O}(n^{53/54})$-approximation algorithm. However, the algorithm uses demand and XOS queries, whereas RepReMatch needs only the weaker value queries.
- Garg et al. [15] offer (fully) polynomial-time approximation schemes and even an optimal algorithm for some types of the symmetric and asymmetric additive NSW.
- For the submodular NSW, Garg et al. [14] devised a family of deterministic algorithms for every $\epsilon > 0$. They are $(4 + \epsilon)$-approximative in the symmetric case and $e(n \cdot \max_{i \in \mathscr{A}}\{\eta_i\} + 2 + \epsilon)$-approximative in the asymmetric one.
- Rado valuations are a special class of submodular functions and stem from a generalisation of OXS valuations. Garg, Kulkarni and Kulkarni emphatically mentioned the lack of research into the OXS NSW. Garg, Husic and Vegh [12] developed an algorithm for the Rado NSW, which is independent from both the number of items and the number of agents. For the symmetric Rado NSW, their algorithm achieves an approximation factor of $256e^{3/e} \approx 772$. For the asymmetric Rado NSW, the approximation factor is $256\gamma^3$ with $\gamma := \max_{i \in \mathscr{A}}\{\eta_i\}/\min_{i \in \mathscr{A}}\{\eta_i\}$. Interestingly, the algorithm is 16γ-approximative in case of the asymmetric additive NSW, so it outperforms both SMatch and RepReMatch in many instances. The algorithm is divided into five phases with purposes related to the ideas of RepReMatch.

Besides efficiency, fairness is also a property towards which algorithm can be designed, although SMatch and RepReMatch advanced little in that regard. Especially RepReMatch with no fairness guarantees is unsatisfactory.

As seen, the research on the NSW gained pace in the last years, and some questions left unanswered by Garg, Kulkarni and Kulkarni have been addressed. Apart from devising more algorithms for other valuation functions or with features close to reality, it could be rewarding to look at more basic cases. For example, as mentioned in Section 1.3, the hardness of the symmetric additive NSW could not be shown to be higher than $\sqrt{8/7} > 1.069$ [11], but the best known algorithm, to the best of our knowledge to this day, has an approximation factor of about 1.45 [3].

References

[1] Arash Asadpour and Amin Saberi. 'An Approximation Algorithm for Max-Min Fair Allocation of Indivisible Goods'. In: *SIAM Journal on Computing* 39.7 (2010), pp. 2970–2989. DOI: 10.1137/080723491.

[2] Tayebeh Bahreini, Hossein Badri and Daniel Grosu. 'An Envy-Free Auction Mechanism for Resource Allocation in Edge Computing Systems'. In: *SEC 2018*. 2018 IEEE/ACM Symposium on Edge Computing (Seattle, WA, USA, 25th–27th Oct. 2018). Ed. by Patrick Kellenberger. The Institute of Electrical and Electronics Engineers, Inc., 2018, pp. 313–322. ISBN: 978-1-5386-9446-6. DOI: 10.1109/SEC.2018.00030. URL: https://www.researchgate.net/publication/329561811_An_Envy-Free_Auction_Mechanism_for_Resource_Allocation_in_Edge_Computing_Systems (visited on 08/06/2023).

[3] Siddharth Barman, Sanath Kumar Krishnamurthy and Rohit Vaish. *Finding Fair and Efficient Allocations*. 11th May 2018. arXiv: 1707.04731 [cs.GT].

[4] Siddharth Barman et al. *Sublinear Approximation Algorithm for Nash Social Welfare with XOS Valuations*. 15th July 2022. arXiv: 2110.00767 [cs.GT].

[5] Ivona Bezáková and Varsha Dani. 'Allocating Indivisible Goods'. In: *ACM SIGecom Exchanges* 5.3 (1st Apr. 2005), pp. 11–18. DOI: 10.1145/1120680.1120683. URL: https://newtraell.cs.uchicago.edu/files/tr_authentic/TR-2004-10.pdf (visited on 11/06/2023).

[6] Ioannis Caragiannis et al. 'The Unreasonable Fairness of Maximum Nash Welfare'. In: *EC '16*. 17th ACM Conference on Economics and Computation (Maastricht, The Netherlands, 24th–28th July 2016). Ed. by Vincent Conitzer, Dirk Bergemann and Yiling Chen. Association for Computing Machinery, 2016, pp. 305–322. ISBN: 978-1-4503-3936-0. DOI: 10.1145/2940716. URL: https://eprints.gla.ac.uk/123283 (visited on 10/06/2023).

[7] Yann Chevaleyre et al. 'Issues in Multiagent Resource Allocation'. In: *Informatica* 30.1 (Jan. 2006), pp. 3–31. URL: https://www.informatica.si/index.php/informatica/article/view/70/62 (visited on 07/06/2023).

[8] Richard Cole and Vasilis Gkatzelis. 'Approximating the Nash Social Welfare with Indivisible Items'. In: *SIAM Journal on Computing* 47.3 (2018), pp. 1211–1236. DOI: 10.1137/15M1053682.

[9] Dagmawi Mulugeta Degefu et al. 'Bankruptcy to Surplus: Sharing Transboundary River Basin's Water under Scarcity'. In: *Water Resources Management* 32 (8 1st June 2018), pp. 2735–2751. ISSN: 1573-1650. DOI: 10.1007/s11269-018-1955-z.

[10] Schahram Dustdar, ed. *NSF Workshop Report on Grand Challenges in Edge Computing*. NSF Edge Computing Workshop (Washington, DC, USA, 26th Oct. 2016). National Science Foundation. 15th Nov. 2017. 17 pp. URL: https://dsg.tuwien.ac.at/team/sd/papers/Bericht_NSF_S_Dustdar.pdf (visited on 08/06/2023).

[11] Jugal Garg, Martin Hoefer and Kurt Mehlhorn. *Satiation in Fisher Markets and Approximation of Nash Social Welfare*. 31st July 2019. arXiv: 1707.04428 [cs.DS].

[12] Jugal Garg, Edin Husic and Laszlo A. Vegh. *Approximating Nash Social Welfare under Rado Valuations*. 30th Sept. 2020. arXiv: 2009.14793 [cs.GT].

[13] Jugal Garg, Pooja Kulkarni and Rucha Kulkarni. *Approximating Nash Social Welfare under Submodular Valuations through (Un)Matchings*. 28th Dec. 2019. arXiv: 1912.12541v1 [cs.GT].

[14] Jugal Garg et al. *Approximating Nash Social Welfare by Matching and Local Search*. 29th Mar. 2023. arXiv: 2211.03883 [cs.GT]. To appear in STOC '23.

[15] Jugal Garg et al. *Tractable Fragments of the Maximum Nash Welfare Problem*. 28th Apr. 2022. arXiv: 2112.10199 [cs.GT].

[16] Subhash Khot and Ashok Kumar Ponnuswami. 'Approximation Algorithms for the Max-Min Allocation Problem'. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007 (Princeton, NJ, USA, 20th–22nd Aug. 2007). Ed. by Moses Charikar et al. Berlin, Heidelberg: Springer, 2007, pp. 204–217. ISBN: 978-3-540-74207-4. DOI: 10.1007/978-3-540-74208-1_15.

[17] Subhash Khot et al. 'Inapproximability Results for Combinatorial Auctions with Submodular Utility Functions'. In: *Algorithmica* 52 (1 13th Oct. 2007), pp. 3–18. ISSN: 0178-4617. DOI: 10.1007/s00453-007-9105-7.

[18] Trung Thanh Nguyen and Jörg Rothe. 'Minimizing envy and maximizing average Nash social welfare in the allocation of indivisible goods'. In: *Discrete Applied Mathematics* 179 (2014), pp. 54–68. ISSN: 0166-218X. DOI: 10.1016/j.dam.2014.09.010.

[19] Hugo Steinhaus. 'The Problem of Fair Division'. In: *Econometrica* 16.1 (Jan. 1948): *Report of the Washington Meeting, September 6-18, 1947*. Ed. by Ragnar Frisch, pp. 101–104. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1914289 (visited on 17/06/2023).

[20] Zoya Svitkina and Lisa Fleischer. *Submodular Approximation: Sampling-based Algorithms and Lower Bounds*. 31st May 2010. arXiv: 0805.1071 [cs.DS].

[21] Jan Vondrák. 'Optimal Approximation for the Submodular Welfare Problem in the Value Oracle Model'. In: *STOC '08*. 40th ACM Symposium on Theory of Computing (Victoria, British Columbia, Canada, 17th–20th May 2008). Ed. by editor. Association for Computing Machinery, 2008, pp. 67–74. ISBN: 978-1-60558-047-0. DOI: 10.1145/3313276.3316304. URL: https://theory.stanford.edu/~jvondrak/data/submod-value.pdf (visited on 10/06/2023).