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

JUGAL GARG, University of Illinois at Urbana-Champaign, USA
POOJA KULKARNI, University of Illinois at Urbana-Champaign, USA
RUCHA KULKARNI, University of Illinois at Urbana-Champaign, USA

We study the problem of approximating maximum Nash social welfare (NSW) when allocating m indivisible items among n asymmetric agents with submodular valuations. The NSW is a well-established notion of fairness and efficiency, defined as the weighted geometric mean of agents' valuations. For special cases of the problem with symmetric agents and additive(-like) valuation functions, approximation algorithms have been designed using approaches customized for these specific settings, and they fail to extend to more general settings. Hence, no approximation algorithm with factor independent of m was known either for asymmetric agents with additive valuations or for symmetric agents beyond additive(-like) valuations before this work.

In this paper, we extend our understanding of the NSW problem to far more general settings. Our main contribution is two approximation algorithms for asymmetric agents with additive and submodular valuations. Both algorithms are simple to understand and involve non-trivial modifications of a greedy repeated matchings approach. Allocations of high-valued items are done separately by un-matching certain items and re-matching them by different processes in both algorithms. We show that these approaches achieve approximation factors of $O(n)$ and $O(n \log n)$ for additive and submodular cases, independent of the number of items. For additive valuations, our algorithm outputs an allocation that also achieves the fairness property of envy-free up to one item (EF1).

Furthermore, we show that the NSW problem under submodular valuations is strictly harder than all currently known settings with an $\frac{e}{e-1}$ factor of the hardness of approximation, even for constantly many agents. For this case, we provide a different approximation algorithm that achieves a factor of $\frac{e}{e-1}$, hence resolving it completely.

CCS Concepts: • **Theory of computation → Approximation algorithms analysis**; **Algorithmic game theory and mechanism design**.

Additional Key Words and Phrases: Nash Social Welfare, Submodular Valuations, Asymmetric Agents

ACM Reference Format:
Jugal Garg, Pooja Kulkarni, and Rucha Kulkarni. 2023. Approximating Nash Social Welfare under Submodular Valuations through (Un)Matchings. *ACM Trans. Algor.* 1, 1 (August 2023), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We study the problem of approximating the maximum Nash social welfare (NSW) when allocating a set $\mathcal{G}$ of m indivisible items among a set $\mathcal{A}$ of n agents with non-negative monotone *submodular* valuations $v_i : 2^{\mathcal{G}} \to \mathbb{R}_+$, and unequal or *asymmetric* entitlements called *agent weights*. Let $\Pi_n(\mathcal{G})$ denote the set of all allocations, i.e., $\{(\mathbf{x}_1, \ldots, \mathbf{x}_n) \mid \cup_i \mathbf{x}_i = \mathcal{G}; \mathbf{x}_i \cap \mathbf{x}_j = \emptyset, \forall i \neq j\}$. The NSW problem is to find an allocation maximizing the weighted geometric mean of valuations,

$$\operatorname*{argmax}_{(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \Pi_n(\mathcal{G})} \left(\prod_{i \in \mathcal{A}} v_i(\mathbf{x}_i)^{\eta_i} \right)^{1/\sum_{i \in \mathcal{A}} \eta_i}, \tag{1}$$

where η_i is the weight of agent i. We call this the *Asymmetric Submodular* NSW *problem*.[1] When agents $\eta_i = 1, \forall i \in \mathcal{A}$, we have the classic NSW problem.

Fair and efficient allocation of resources is a central problem in economic theory. The NSW objective provides an interesting trade-off between the two extremal objectives of social welfare (i.e., sum of valuations) and max-min fairness, and in contrast to both, it is invariant to individual scaling of each agent's valuations (see [36] for additional characteristics). It was independently discovered by three different communities as a solution to the bargaining problem in classic game theory [37], a well-studied notion of proportional fairness in networking [29], and coincides with the celebrated notion of competitive equilibrium with equal incomes (CEEI) in economics [42]. While Nash [37] only considered the symmetric case, Harsanyi and Selten [27] and Kalai [28] proposed the asymmetric case, which has also been extensively studied, and used in many different applications, e.g., bargaining theory [12, 32, 41], water allocation [20, 26], climate agreements [44], and many more.

The NSW problem is known to be notoriously hard, e.g., NP-hard even for two agents with identical additive valuations, and APX-hard in general [33].[2] Effort was then devoted to develop efficient approximation algorithms. A series of remarkable works [1, 2, 8, 15–17, 21] provide good approximation guarantees for the special subclasses of this problem where agents are symmetric and have additive(-like) valuation functions[3] via utilizing ingenious different approaches. All these approaches exploit the symmetry of agents and the characteristics of additive-like valuation functions,[4] which makes them hard to extend to the asymmetric case and more general valuation functions. As a consequence, no approximation algorithm with a factor independent of the number of items m [38] was known either for asymmetric agents with additive valuations or for symmetric agents beyond additive(-like) valuations before this work. These questions are also raised in [8, 16].

The NSW objective also serves as a major focal point in fair division. For the case of symmetric agents with additive valuations, Caragiannis et al. [11] present a compelling argument in favor of the 'unreasonable' fairness of maximum NSW by showing that such an allocation has outstanding properties, namely, it is EF1 (a popular fairness property of envy-freeness up to one item) as well as Pareto optimal (PO), a standard notion of economic efficiency. Even though computing a maximum NSW allocation is hard, its approximation recovers most of the fairness and efficiency guarantees; see, e.g., [8, 15, 25].

---

[1]In the rest of this paper, we refer to various special cases of the problem as the α μ NSW problem, where α is the nature of agents, symmetric or asymmetric, and μ is the type of agent valuation functions. We skip one or both qualifiers when they are clear from the context.

[2]Observe that *the partition problem* reduces to the NSW problem with two identical agents.

[3]Slight generalizations of additive valuations are studied: budget additive [21], separable piecewise linear concave (SPLC) [2], and their combination [15].

[4]For instance, the notion of a maximum bang-per-buck (MBB) item is critically used in most of these approaches. There is no such equivalent notion for the submodular case.

In this paper, we extend our understanding of the NSW problem to far more general settings. Our main contribution is two approximation algorithms, SMatch and RepReMatch, for asymmetric agents with additive and submodular valuations, respectively. Both algorithms are simple to understand and involve non-trivial modifications of a greedy repeated matchings approach. Allocations of high-valued items are done separately by un-matching certain items and re-matching them by different processes in both algorithms. We show that these approaches achieve approximation factors of $O(n)$ and $O(n \log n)$ for additive and submodular cases, independent of the number of items. Our algorithm outputs an allocation that is also EF1 for additive valuations.

1.1 Model

We formally define the valuation functions we consider in this paper and their relations to other commonly-studied valuation functions. For convenience, we also use $v_i(j)$ instead of $v_i(\{j\})$ to denote the valuation of agent i for item j.

(1) Additive: Given the valuation $v_i(j)$ of each agent i for every item j, the valuation for a set of items is the sum of the individual valuations. That is, $\forall S \subseteq G, v_i(S) = \sum_{j \in S} v_i(j)$. In the restricted additive case, $v_i(j) = \{0, v_j\}, \forall i$.

(2) Budget additive (BA): Every agent has an upper cap on the maximum valuation she can receive from any allocation. For any set of items, the agent's total valuation is the minimum value from the additive value of this set and the cap. i.e., $\forall S \subseteq G, v_i(S) = \min\{\sum_{j \in S} v_i(j), c_i\}$, where c_i denotes agent i's cap.

(3) Separable piecewise linear concave (SPLC): In this case, there are multiple copies of each item. An agent's valuation is piecewise linear concave for each item, and it is additively separable across items. Let $v_i(j, k)$ denote the agent i's value for receiving k^{th} copy of item j. Concavity implies that $v_i(j, 1) \geq v_i(j, 2) \ldots, \forall i, j$. The valuation of agent i for a set S of items, containing l_j copies of items j, is $v_i(S) = \sum_j \sum_{k=1}^{l_j} v_i(j, k)$.

(4) Monotone Submodular: Let $v_i(S_1 \mid S_2)$ denote the marginal utility of agent i for a set S_1 of items over set S_2, where $S_1, S_2 \subseteq G$ and $S_1 \cap S_2 = \emptyset$. Then, the valuation function of every agent is a monotonically non-decreasing function $v_i : 2^G \to \mathbb{R}_+$ that satisfies the submodularity constraint that for all $i \in \mathcal{A}, h \in G, S_1, S_2 \subseteq G$,

$$v_i(h \mid S_1 \cup S_2) \leq v_i(h \mid S_1).$$

Other popular valuation functions are OXS, gross substitutes (GS), XOS and subadditive [39]. These function classes are related as follows:

$$\text{Additive} \subsetneq \begin{matrix} \text{SPLC} \subsetneq \text{OXS} \\ \text{BA} \end{matrix} \subsetneq \text{GS} \subsetneq \text{Submodular} \subsetneq \text{XOS} \subsetneq \text{subadditive}.$$

1.2 Results

Table 1 summarizes approximation guarantees of the algorithms RepReMatch and SMatch under popular valuation functions, formally defined in Section 1.1. Here, the approximation guarantee of an algorithm is defined as α for an $\alpha \geq 1$, if it outputs an allocation whose (weighted) geometric mean is at least $1/\alpha$ times the maximum (optimal) geometric mean. All current best-known results are also stated in the table for reference.

To complement these results, we also provide a $\frac{e}{e-1} = 1.5819$-factor hardness of approximation result for the submodular NSW problem in Section 4. This hardness even applies when the number of agents is constant. This shows that the general problem is strictly harder than the settings studied so far, for which 1.45 factor approximation algorithms are known.

Valuations	Symmetric Agents		Asymmetric Agents	
	NP-Hardness	Algorithm	NP-Hardness	Algorithm
Restricted Additive	1.069 [21]	1.45[8] [S]	1.069 [21]	$O(n)$ [S]
Additive	—"—	1.45 [8]	—"—	—"—
Budget additive SPLC	—"—	1.45 [15]	—"—	—"—
OXS Gross substitutes	—"—	$O(n \log n)$ [R]	—"—	$O(n \log n)$ [R]
Submodular	1.5819 [Thm 4.1]	—"—	1.5819 [Thm 4.1]	—"—
XOS Subadditive	—"—	$O(m)$ [38]	—"—	$O(m)$ [38]

Table 1. Summary of results. Every entry has the best-known approximation guarantee for the setting followed by the reference that establishes it from this paper or otherwise. Here, [S] and [R] respectively refer to Algorithms SMatch and RepReMatch.

For the special case of the submodular NSW problem where the number of agents is constant, we describe another algorithm with a *matching* 1.5819 *approximation factor* in Section 5, hence resolving this case completely. Finally, in the same section, we show that for the symmetric additive NSW problem, the allocation of items returned by SMatch also satisfies EF1. Finally, a 1.45-factor guarantee can be shown for the further special case of restricted additive valuations, showing that the allocation returned by the algorithm is PO. This matches the current best-known approximation factor for this case.

1.3 Techniques

We describe the techniques used in this work in a pedagogical manner. We start with a naive algorithm and build progressively more sophisticated algorithms by fixing the main issues that result in bad approximation factors for the corresponding algorithms, finally ending with our algorithms.

All approaches compute, sometimes multiple, maximum weight matchings of weighted bipartite graphs. These graphs have agents and items in separate parts, and the edge weight assigned for matching an item j to an agent i is the logarithm of the valuation of the agent for the item, scaled by the agent's weight, i.e., $\eta_i \log v_i(j)$. Observe that, by taking the logarithm of the NSW objective (1), we get an equivalent problem where the objective is to maximize the weighted sum of logarithms of agents' valuations.

Let us first consider the additive NSW problem and see what NSW is assured by computing a single such maximum weight matching. If the number of agents, say n, and items, say m, is the same, then the allocation obtained by matching items to agents according to such a matching results in the maximum NSW objective. Nguyen and Rothe [38] extend this algorithm to the general case, by allocating n items according to one matching and arbitrarily allocating the remaining items. They prove that this gives an $(m - n + 1)$−factor approximation algorithm.

A natural extension to this algorithm is to compute more matchings instead of arbitrary allocations after a single matching. That is, compute one maximum weight matching, allocate items according to this matching, then repeat this process until all items are allocated. This repeated matching algorithm still does not help us eliminate the dependence on m in the approximation factor. To see why, consider the following example.

Example 1.1. Consider 2 agents A, B with weights 1 each, and $m + 1$ items. The valuations of A and B for the first item are $M + \epsilon$ and M, respectively. Agent A values each of the remaining items at 1, while B only values the last of these at 1 and the remaining $(m - 1)$ items at 0. An allocation that optimizes the NSW of the agents allocates the first item to B and all remaining items to A. The optimal geometric mean is $(Mm)^{1/2}$. A repeated matching algorithm allocates the first item to A and the last to B in the first iteration. No matching can now give additional utility to B. The maximum geometric mean that can be generated by such an algorithm is $((M + \epsilon + m - 1)1)^{1/2} < \sqrt{M + m}$. Thus, using $M := m$, the ratio of these two geometric means depends on m.

The above example shows the critical reason why a naive repeated matching algorithm may not work. In the initial matchings, the algorithm has no knowledge of how the agents value the entire set of items. Hence during these matching, it might allocate the high-valued items to the wrong agents, thereby reducing the NSW by a large factor. To get around this problem, our algorithm needs to have some knowledge of an agent's valuation for the unallocated (low-valued) set of items while deciding how to allocate high-valued items. It can then allocate the high-valued items correctly with this foresight.

It turns out that there is a simple way to provide this foresight when the valuation functions are additive(-like). Effectively, we keep aside $O(n)$ high-valued items of each agent and assign the other items via a repeated matching algorithm. We then assign the items we had set aside to all agents via matchings that locally maximize the resulting NSW objective. The collective set of items put aside by all agents will have all the high-valued items that require the foresight for correct allocation as a subset. Because these items are allocated after allocating the low-valued items, this algorithm allocates the high-valued items more smartly. In Section 2, we describe this algorithm, termed SMatch, and show that it gives an $O(n)$ factor approximation for the NSW objective.

The above idea, however, does not work for submodular valuation functions. The reason is subtle and is described as follows. Even in the additive case, the idea requires keeping aside not the set of items with the highest valuation but the set of items that leave a set of lowest valuation. For additive valuations, these sets are the same. However, it is known from [40] that finding a set of items of minimum valuation with lower bounded cardinality for monotone submodular functions is inapproximable within $\sqrt{m/\log m}$ factor, where m is the number of items.

We get around this issue by assigning high-valued items differently. Interestingly, we once again use the repeated matching technique for this. In the algorithm RepReMatch, we allocate items via repeated matchings, then release some of the initial matchings and re-match the items of these initial matchings.

The idea is that the initial matchings will allocate all high-valued items, even if incorrectly, and give us the set of items that must be allocated correctly. If the total number of all high-valued items depends only on n, then the problem of maximizing the NSW objective when allocating this set of items is solved up to some factor of n by applying a repeated matching algorithm. In Lemma 3.5, we prove such an upper bound on the number of initial matchings to be released.

Thus far, we have proved that we can allocate one set of items, the high-valued items, approximately optimally. Now, submodular valuations do not allow us to add valuations incurred in separate matchings to compute the total valuation of an agent. Therefore, if we want a repeated matching-like technique to work, we require the following natural modification. We redefine the edge weights used for computing matchings. We now consider the marginal valuations over items already allocated in previous matchings as edge weights rather than individual valuations.

There are several challenges in proving that this approach gives an allocation with a high NSW overall. First, bounding the amount of valuation received by a particular agent as a fraction of her optimal allocation is difficult. This is because the subset of items the algorithm allocates might

differ completely from the set of optimal items. We can, however, give a relation between these two values, and this is seen in Lemma 3.4.

Then, since we release and reallocate the items of initial matchings, the set of items allocated to an agent can be completely different from the set before, changing all marginal utilities completely. It is thus non-trivial to combine the valuations from these stages too. This is done in the proof of Theorem 3.1.

Apart from this, we also have the following results in the paper that use different techniques.

Submodular NSW **with constant number of agents.** We completely resolve this case using a different approach that uses techniques of maximizing submodular functions over matroids developed in [14] and a reduction from [43]. At a high level, we first maximize the continuous relaxations of agent valuation functions, then round them using a randomized algorithm to obtain an integral allocation of items. The two key results used in designing the algorithm are Theorems 5.3 and 5.4.

Hardness of approximation. The submodular ALLOCATION problem is to maximize the sum of valuations of agents over integral allocations of items. Khot et al. [30] described a reduction of MAX-3-COLORING, which is NP-Hard to approximate within a constant factor, to ALLOCATION. We prove that this reduction also establishes the same hardness for the submodular NSW problem.

1.4 Further Related Work

An extensive work has been done on special cases of the NSW problem. Several constant-factor approximation algorithms have been obtained for the symmetric additive NSW problem. The first such algorithm used an approach based on a variant of Fisher markets [17] to achieve an approximation factor of $2 \cdot e^{1/e} \approx 2.889$. Later, the analysis of this algorithm was improved to 2 [16]. Another approach based on the theory of real stable polynomials gave an e-factor guarantee [1]. Recently, Barman et al. [8] obtained the current best approximation factor of $e^{1/e} \approx 1.45$ using an approach based on approximate EF1 and PO allocation. These approaches have also been extended to provide constant-factor approximation algorithms for slight generalizations of additive valuations, namely the budget additive [21], SPLC [2], and a common generalization of these two valuations [15].

All these approaches exploit the symmetry of agents and the characteristics of additive-like valuation functions. For instance, the notion of a maximum bang-per-buck (MBB) item is critically used in most of these approaches. There is no such equivalent notion for the submodular case. This makes them hard to extend to the asymmetric case and to more general valuation functions.

Fair and efficient division of items to asymmetric agents with submodular valuations is an important problem, also raised in [16]. The asymmetry is a reflection of the difference in the entitlements of the agents. Submodularity models the concept of diminishing marginal valuations. It is reasonable to assume that both these characteristics model real-world scenarios better than the commonly studied albeit important setting of symmetric agents with additive valuations. However, the only known result for this general problem is an $O(m)$-factor algorithm [38], where m is the number of items.

Two other popular welfare objectives are utilitarian and egalitarian welfare. In social welfare, the goal is to maximize the sum of valuations of all agents, and in the max-min objective, the goal is to maximize the value of the lowest-valued agent. The latter objective is also termed the Santa Claus problem for the restricted additive valuations [5].

The social welfare problem under submodular valuations has been completely resolved with a $\frac{e}{e-1} = 1.5819$-factor algorithm [43] and a matching hardness result [30]. Note that the additive case

for this problem has a trivial linear time algorithm, hence it is perhaps unsurprising that a constant factor algorithm would exist for the submodular case.

Extensive work has been done on the restricted additive valuations case for the max-min objective, resulting in constant factor algorithms for the same [3, 19]. However, for the unrestricted additive valuations, the best approximation factor is $O(\sqrt{n}\log^3 n)$ [4]. For the submodular Santa Claus problem, there is an $O(n)$ factor algorithm [31]. On the other hand, a hardness factor of 2 is the best-known lower bound for both settings [9].

1.5 Subsequent Work

After this paper was first published [24], there has been a series of works in the area of Nash social welfare for beyond-additive valuation functions, many of them extending the novel approach of this paper. Barman et al. [6] and Chaudhury et al. [13] showed $O(n)$-approximation algorithm when valuations of agents are subadditive. [6] also showed that one cannot get better than $O(n)$ factor approximation when we restrict access to the functions via value oracles. This hardness result is a query complexity result, not conditioned on complexity-theoretic assumptions. Garg et al. [23] developed the first constant factor approximation algorithm for Nash social welfare for a broad class of gross substitutes valuations that strictly subsumed SPLC valuations, which they called *Rado* valuations. They gave a 772-approximation for approximating NSW with symmetric agents and a $256\gamma^3$ approximation for NSW with asymmetric agents, where γ is the ratio of the largest to the smallest weight of an agent. Li and Vondrák [35] extended this work to show a 380-approximation for agents with submodular valuations building on top of [23]. Finally, in another breakthrough work, Garg et al. [22] gave a $(4 + \varepsilon)$-approximation for NSW under submodular valuations with symmetric agents for any $\varepsilon > 0$ and $(\gamma + 2 + \epsilon)$-approximation with asymmetric agents. For valuations beyond submodular, Barman et al. [7] gave a sublinear approximation for XOS valuations with symmetric agents.

Organization of the paper: In Section 2, we describe the algorithm SMatch and analysis for the additive NSW problem. In Section 3, we present the algorithm RepReMatch for submodular valuations. Section 4 contains the hardness proof for the submodular setting. The results for the special cases of submodular NSW with constant number of agents, symmetric additive NSW, and symmetric additive NSW with restricted valuations are presented in Section 5. In Section 6, we present counter-examples to prove the tightness of the analysis of Algorithms RepReMatch and SMatch. The final Section 7 discusses possible further directions.

2 ADDITIVE VALUATIONS

In this section, we present SMatch, described in Algorithm 1, for the asymmetric additive NSW problem and prove the following approximation result.

THEOREM 2.1. *The* NSW *objective of allocation* **x**, *output by* SMatch *for asymmetric additive* NSW *problem, is at least* $1/2n$ *times the optimal* NSW, *denoted as* OPT, *i.e.,* $\text{NSW}(\mathbf{x}) \geq \frac{1}{2n}\text{OPT}$.

SMatch is a single pass algorithm that allocates up to one item to every agent per iteration to maximize the NSW objective locally. An issue with a naive single pass, locally optimizing greedy approach is that the initial iterations work on highly limited information. Example 1.1 shows that such algorithms can result in outcomes with very low NSW even for symmetric agents with additive valuation functions. In the example, although agent A can be allocated an item of high valuation later, the algorithm does not *know* this initially. Algorithm 1 resolves this issue by pre-computing an approximate value that the agents will receive in later iterations, and uses this information in the edge weight definitions when allocating the first item to every agent. We now discuss the details of SMatch.

2.1 Algorithm

SMatch works in a single pass. For every agent, the algorithm first computes the value of $m - 2n$ least valued items and stores this in u_i. SMatch then defines a weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}, \mathcal{W})$ with edge weights $w(i, j) = \eta_i \log \left(v_i(j) + \frac{u_i}{n} \right)$, and allocates one item to each agent along the edges of a maximum weight matching of Γ. It then starts allocating items via repeated matchings. Until all items are allocated, SMatch iteratively defines graphs $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with $\mathcal{G}^{rem}$ denoting the set of unallocated items and edge weights defined as $w(i, j) = \eta_i \log \left(v_i + v_i(j) \right)$, where v_i is the valuation of agent i for items that are allocated to her. SMatch then allocates at most one item to each agent according to a maximum weight matching of Γ.

Algorithm 1: SMatch for the Asymmetric Additive NSW problem

Input : A set $\mathcal{A}$ of n agents with weights η_i, $\forall i \in \mathcal{A}$, a set $\mathcal{G}$ of m indivisible items, and additive valuations $v_i : 2^{\mathcal{G}} \rightarrow \mathbb{R}_+$, where $v_i(\mathcal{S})$ is the valuation of agent $i \in \mathcal{A}$ for a set of items $\mathcal{S} \subseteq \mathcal{G}$.

Output: An allocation that approximately optimizes the NSW.

1 $\mathbf{x}_i \leftarrow \emptyset, u_i \leftarrow v_i(\mathcal{G}_{i,[2n+1:m]})$ $\forall i \in [n]$ `// ` $\mathcal{G}_{i,[a:b]}$ ` defined in Section 2.2`
2 Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}, \mathcal{W})$ with weights
   $\mathcal{W} = \{ w(i, j) \mid w(i, j) = \eta_i \log \left(v_i(j) + \frac{u_i}{n} \right), \forall i \in \mathcal{A}, j \in \mathcal{G} \}$
3 Compute a maximum weight matching $\mathcal{M}$ for Γ
4 $\mathbf{x}_i \leftarrow \mathbf{x}_i \cup \{ j \mid (i, j) \in \mathcal{M} \}, \forall i \in \mathcal{A}$ `// allocate items according to ` $\mathcal{M}$
5 $\mathcal{G}^{rem} \leftarrow \mathcal{G} \setminus \{ j \mid (i, j) \in \mathcal{M} \}$ `// update set of unallocated items`

6 **while** $\mathcal{G}^{rem} \neq \emptyset$ **do**
7 $\quad$ Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with weights
   $\quad \mathcal{W} = \{ w(i, j) \mid w(i, j) = \eta_i \log(v_i(j) + v_i(\mathbf{x}_i)), \forall i \in \mathcal{A}, j \in \mathcal{G}^{rem} \}$
8 $\quad$ Compute a maximum weight matching $\mathcal{M}$ for Γ
9 $\quad \mathbf{x}_i \leftarrow \mathbf{x}_i \cup \{ j \mid (i, j) \in \mathcal{M} \}, \forall i \in \mathcal{A}$ `// allocate items according to ` $\mathcal{M}$
10 $\quad \mathcal{G}^{rem} \leftarrow \mathcal{G}^{rem} \setminus \{ j \mid (i, j) \in \mathcal{M} \}$ `// remove allocated items`
11 **end**
12 Return $\mathbf{x}$

REMARK 2.1. *In Algorithm SMatch [1], we can also use a round-robin allocation instead of repeated matchings in lines 6 to 11. Similarly, in Algorithm RepReMatch [2], we can use round-robin allocation instead of repeated matching in Phase 2. This will improve the running time of both algorithms. However, we stick with repeated matchings in our algorithms because the idea of repeated matchings has become a useful tool in subsequent works on NSW approximation.*

2.2 Notation

In the following discussion, we use $\mathbf{x}_i = \{ h_i^1, \ldots, h_i^{\tau_i} \}$ to denote the set of τ_i items received by agent i in SMatch. We use $\mathbf{x}_i^* = \{ g_i^1, \ldots, g_i^{\tau_i^*} \}$ to denote the set of τ_i^* items in i's optimal bundle. Then for every i, all items in $\mathbf{x}_i$ and $\mathcal{G}$ are ranked according to the decreasing utilities as per v_i. We use the shorthand $[n]$ to denote the set $\{ 1, 2, \ldots, n \}$. Let $\mathcal{G}_{i,[a:b]}$ denotes the items ranked from a to b according to agent i in $\mathcal{G}$, and $\mathbf{x}_{i,1:t}$ is the total allocation to agent i from the first t matching iterations. We also use $\mathcal{G}_{i,k}$ to denote the kth ranked item of agent i from the entire set of items.

For all i, we define u_i as the minimum value for the remaining set of items upon removing at most $2n$ items from $\mathcal{G}$, i.e., $u_i = \min_{\mathcal{S} \subseteq \mathcal{G}, |\mathcal{S}| \le 2n} v_i(\mathcal{G} \setminus \mathcal{S}) = \mathcal{G}_{i,[2n+1,m]}$.[5]

2.3 Analysis

To establish the guarantee of Theorem 2.1, we first prove a couple of lemmas.

LEMMA 2.2. $v_i(h_i^t) \ge v_i(\mathcal{G}_{i,tn})$.

PROOF. Since every iteration of SMatch allocates at most n items, at the start of iteration t at most $(t-1)n$ items are allocated. Thus at least n items from $\mathcal{G}$ ranked between 1 to tn by agent i are still unallocated. In the tth iteration the agent will thus get an item with value at least the value of $\mathcal{G}_{i,tn}$ and the lemma follows.   □

LEMMA 2.3. $v_i(h_i^2, \ldots, h_i^{\tau_i}) \ge \frac{u_i}{n}$.

PROOF. Using Lemma 2.2 and since $v_i(\mathcal{G}_{i,tn}) \ge v_i(\mathcal{G}_{i,tn+k})$, $\forall k \in [n-1]$

$$v_i(h_i^t) \ge \frac{1}{n} v_i(\mathcal{G}_{i,[tn:(t+1)n-1]}) \ .$$

Thus,

$$v_i(h_i^2, \ldots, h_i^{\tau_i}) = \sum_{t=2}^{\tau_i} v_i(h_i^t) \ge \frac{1}{n} \sum_{t=2}^{\tau_i} v_i(\mathcal{G}_{i,[tn:(t+1)n-1]}) \ .$$

As at most n items are allocated in every iteration, agent i receives items for at least $\lfloor \frac{m}{n} \rfloor$ iterations.[6] This implies that $(\tau_i + 1)n \ge m$ and hence,

$$v_i(h_i^2, \ldots, h_i^{\tau_i}) \ge \frac{1}{n} \left(v_i(\mathcal{G}_{i,[2n:m-1]}) \right)$$

$$\ge \frac{1}{n} \left(v_i(\mathcal{G}_{i,[2n+1:m]}) \right) = \frac{1}{n} u_i \ .$$

The second inequality follows as $v_i(\mathcal{G}_{i,2n}) \ge v_i(\mathcal{G}_{i,m})$.   □

We now prove the main theorem.

PROOF OF THEOREM 2.1.

$$\text{NSW}(\mathbf{x}) = \prod_{i=1}^n \left(v_i(h_i^1, \ldots, h_i^{\tau_i})^{\eta_i} \right)^{\frac{1}{\sum_{i=1}^n \eta_i}}$$

$$= \prod_{i=1}^n \left(\left(v_i(h_i^1) + v_i(h_i^2 \ldots, h_i^{\tau_i}) \right)^{\eta_i} \right)^{\frac{1}{\sum_{i=1}^n \eta_i}}$$

$$\ge \prod_{i=1}^n \left(\left(v_i(h_i^1) + \frac{u_i}{n} \right)^{\eta_i} \right)^{\frac{1}{\sum_{i=1}^n \eta_i}} ,$$

where the last inequality follows from Lemma 2.3. During the allocation of the first item h_i^1, items g_i^1 of all agents are available. Thus, allocating each agent her own g_i^1 is a feasible first matching and we get

$$\text{NSW}(\mathbf{x}) \ge \prod_{i=1}^n \left(\left(v_i(g_i^1) + \frac{u_i}{n} \right)^{\eta_i} \right)^{\frac{1}{\sum_{i=1}^n \eta_i}} \ .$$

---

[5]As the valuation functions are monotone, the minimum value will be obtained by removing exactly $2n$ items. The *less than* accounts for the case when the number of items in $\mathcal{G}$ is fewer than $2n$.

[6]Here we assume that the agents have a non-zero valuation for every item. If it does not, the other case is also straightforward and the lemma continues to hold.

Now, $u_i = \min_{\mathcal{S} \subseteq \mathcal{G}, |\mathcal{S}| \le 2n} v_i(\mathcal{G} \setminus \mathcal{S})$. Suppose we define, $\mathcal{S}_i^* = \arg \min_{|\mathcal{S}| \le 2n, \mathcal{S} \subseteq \mathbf{x}_i^*} v_i(\mathbf{x}_i^* \setminus \mathcal{S})$, then $v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*) \le u_i$. It follows by using $\mathcal{S}_i = \arg \min_{\mathcal{S} \subseteq \mathcal{G}, |\mathcal{S}| \le 2n} v_i(\mathcal{G} \setminus \mathcal{S})$, we get $u_i = v_i(\mathcal{G} \setminus \mathcal{S}_i) \ge v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i) \ge v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*)$. Thus,

$$\mathrm{NSW}(\mathbf{x}) \ge \prod_{i=1}^{n} \left(\left(\frac{1}{2n} v_i(\mathcal{S}_i^*) + \frac{1}{n} v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*) \right)^{\eta_i} \right)^{\frac{1}{\sum_i \eta_i}}$$

$$\ge \frac{1}{2n} \prod_{i=1}^{n} \left(v_i(\mathbf{x}_i^*)^{\eta_i} \right)^{\frac{1}{\sum_{i=1}^{n} \eta_i}} = \frac{1}{2n} \mathrm{OPT} \ . \qquad \square$$

REMARK 2.2. *When* SMatch *is applied to the instance of Example 1.1, it results in a better allocation than that of a naive repeated matching approach. Stage 1 of* SMatch *computes u_i as $m - 2n$ and 0 for A and B respectively. When this value is included in the edge weight of the first bipartite graph Γ, the resulting matching gives B the first item, and A some other item. Subsequently, A gets all remaining items, resulting in an allocation having the optimal* NSW.

The algorithm SMatch easily extends to budget additive (BA) and separable piecewise concave (SPLC) valuations using the following small changes: In BA, $u_i := \min(c_i, \mathcal{G}_{1,[2n+1:m]})$ where c_i is the utility cap for agent i, and in SPLC, u_i needs to be calculated while considering each copy of an item as a separate item. In both cases, the edge weights in the bipartite graphs will use a marginal utility (as we use in the case of the submodular valuation in Section 3). Lemma 2.3 and the subsequent proofs can be easily extended for these cases by combining ideas from Lemma 3.4 and Proof of Theorem 3.1. Thus, we obtain the following theorem.

THEOREM 2.4. *The* NSW *objective of allocation $\mathbf{x}$, output by* SMatch *for asymmetric* BA *(and* SPLC*)* NSW *problem, is at least $1/2n$ times the optimal* NSW*, denoted as* OPT*, i.e.,* $\mathrm{NSW}(\mathbf{x}) \ge \frac{1}{2n} \mathrm{OPT}$.

3 SUBMODULAR VALUATIONS

In this section, we present the RepReMatch, given in Algorithm 2, for approximating the NSW objective under submodular valuations. We will prove the following relation between the NSW of the allocation $\mathbf{x}$ returned by RepReMatch and the optimal geometric mean OPT.

THEOREM 3.1. *The* NSW *objective of allocation $\mathbf{x}$, output by* RepReMatch *for asymmetric submodular* NSW *problem, is at least $1/2n(\log n + 2)$ times the optimal* NSW*, denoted as* OPT*, i.e.,* $\mathrm{NSW}(\mathbf{x}) \ge \frac{1}{2n(\log n + 2)} \mathrm{OPT}$.

3.1 Algorithm

RepReMatch takes as input an instance of the NSW problem, denoted by $(\mathcal{A}, \mathcal{G}, \mathcal{V})$, where $\mathcal{A}$ is the set of agents, $\mathcal{G}$ is the set of items, and $\mathcal{V} = \{v_1, v_2 \ldots, v_n\}$ is the set of agents' monotone submodular valuation functions, and generates an allocation vector $\mathbf{x}$. Each agent $i \in \mathcal{A}$ is associated with a positive weight η_i.

RepReMatch runs in three phases. In the first phase, in every iteration, we define a weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ as follows. $\mathcal{G}^{rem}$ is the set of items that are still unallocated ($\mathcal{G}^{rem} = \mathcal{G}$ initially). The weight of edge $(i, j), i \in \mathcal{A}, j \in \mathcal{G}^{rem}$, denoted by $w(i, j) \in \mathcal{W}$, is defined as the logarithm of the valuation of the agent for the singleton set having this item, scaled by the agent's weight. That is, $w(i, j) = \eta_i \log(v_i(j))$. We then compute a maximum weight matching in this graph and allocate to agents the items they were matched to (if any). This process is repeated for $\lceil \log n \rceil$ iterations.

In the second phase, we perform a similar repeated matching process with different edge weight definitions for the graphs Γ. We start this phase by assigning empty bundles to all agents. Here, the

weight of an edge between agent i and item j is defined as the logarithm of the valuation of agent i for the set of items currently allocated to her in Phase 2 of RepReMatch, scaled by her weight. That is, if we denote the items allocated in t iterations of Phase 2 as $\mathbf{x}_{i,t}^2$, in $(t+1)^{st}$ iteration, $w(i,j) = \eta_i \log(v_i(\mathbf{x}_{i,t}^2 \cup \{j\}))$.

In the final phase, we re-match the items allocated in Phase 1. We release these items from their agents and define $\mathcal{G}^{rem}$ as the union of these items. We define Γ by letting the edge weights reflect the total valuation of the agent upon receiving the corresponding item, i.e., $w(i,j) = \eta_i \log(v_i(\mathbf{x}_i^2 \cup \{j\}))$, where $\mathbf{x}_i^2$ is the final set of items allocated to i in Phase 2. We compute one maximum weight matching for Γ so defined and allocate all items along the matched edges. All remaining items are then arbitrarily allocated. The final allocations to all agents, denoted as $\mathbf{x} = \{\mathbf{x}_i\}_{i \in \mathcal{A}}$, is the output of RepReMatch.

3.2 Notation

There are three phases in RepReMatch. We denote the set of items received by agent i in Phase $p \in \{1, 2, 3\}$ by $\mathbf{x}_i^p$, and its size $|\mathbf{x}_i^p|$ by τ_i^p. Similarly, $\mathbf{x}_i$ and τ_i denote the final set of items received by agent i and the size of this set. Note that Phase 3 releases and re-allocates selected items of Phase 1, thus τ_i is not equal to $\tau_i^1 + \tau_i^2 + \tau_i^3$. The items allocated to the agents in Phase 2 are denoted by $\mathbf{x}_i^2 = \{h_i^1, h_i^2 \ldots, h_i^{\tau_i^2}\}$. We also refer to the complete set of items received in iterations 1 to t of Phase p by $\mathbf{x}_{i,t}^p$, for any $p \in \{1, 2, 3\}$.

For the analysis, the marginal utility of an agent i for an item j over a set of items $\mathcal{S}$ is denoted by $v_i(j \mid \mathcal{S}) = v_i(\{j\} \cup \mathcal{S}) - v_i(\mathcal{S})$. Similarly, we denote by $v_i(\mathcal{S}_1 \mid \mathcal{S}_2)$ the marginal utility of set $\mathcal{S}_1$ of items over set $\mathcal{S}_2$ where $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{G}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. We use $\mathbf{x}^* = \{\mathbf{x}_i^* \mid i \in \mathcal{A}\}$ to denote the optimal allocation of all items that maximize the NSW, and τ_i^* for $|\mathbf{x}_i^*|$. For every agent i, items in $\mathbf{x}_i^*$ are ranked so that g_i^j is the item that gives i the highest marginal utility over all higher-ranked items. That is, for $j = 1$, g_i^1 is the item that gives i the highest marginal utility over $\emptyset$, and for all $2 \le j \le \tau_i^*$, $g_i^j = \mathrm{argmax}_{g \in \mathbf{x}_i^* \setminus \{g_i^1, \ldots, g_i^{j-1}\}} v_i(g \mid \{g_i^1, \ldots, g_i^{j-1}\})$.[7]

We let $\bar{\mathbf{x}}_i^*$ denote the set of items from $\mathbf{x}_i^*$ that are not allocated (to any agent) at the end of Phase 1, and we denote by $\bar{v}_i^* = v_i(\bar{\mathbf{x}}_i^*)$ and $\bar{\tau}_i^* = |\bar{\mathbf{x}}_i^*|$ respectively the total valuation and number of these items. For convenience, to specify the valuation for a set of items $\mathcal{S}_1 = \{s_1^1, \ldots s_1^{k_1}\}$, instead of $v_i(\{s_1^1, \ldots, s_1^{k_1}\})$, we also use $v_i(s_1^1, \ldots, s_1^{k_1})$. Similarly, while defining the marginal utility of a set $\mathcal{S}_2 = \{s_2^1, \ldots, s_2^{k_2}\}$ over $\mathcal{S}_1$ instead of writing $v_i(\{s_2^1, \ldots, s_2^{k_2}\} \mid \{s_1^1, \ldots, s_1^{k_1}\})$, we also use $v_i(s_2^1, \ldots, s_2^{k_2} \mid s_1^1, \ldots, s_1^{k_1})$.

3.3 Analysis

We will prove Theorem 3.1 using a series of supporting lemmas. We first prove that in Phase 2, the minimum marginal utility of an item allocated to an agent over her current allocation from previous iterations of Phase 2 is not too small. This is the main result that allows us to bound the minimum valuation of the items allocated in Phase 2.

In the tth iteration of Phase 2, RepReMatch finds a maximum weight matching. Here, the algorithm tries to assign to each agent an item that gives her the maximum marginal utility over her currently allocated set of items. However, every agent is competing with $n - 1$ other agents to get this item. So, instead of receiving the best item, she might lose a few high-ranked items to other agents. Suppose in t^{th} iteration, the set of goods allocated (across all agents) is $\mathcal{S}^t$. The goods that

---

[7]Since the valuations are monotone submodular, this ensures that $v_i(g_i^j \mid \{g_i^1, \ldots, g_i^{j-1}\}) \ge v_i(g_i^k \mid \{g_i^1, \ldots, g_i^{k-1}\})$ for all $k \ge j$. This implies that in any subset of ℓ items in the optimal bundle, the highest ranked item's marginal contribution is at least $1/\ell$ times that of this set, when the marginal contribution is counted in this way.

Algorithm 2: RepReMatch for the Asymmetric Submodular NSW problem

---

Input : A set $\mathcal{A}$ of n agents with weights η_i, $\forall i \in \mathcal{A}$, a set $\mathcal{G}$ of m indivisible items, and valuations $v_i : 2^{\mathcal{G}} \to \mathbb{R}_+$, where $v_i(\mathcal{S})$ is the valuation of agent $i \in \mathcal{A}$ for a set of items $\mathcal{S} \subseteq \mathcal{G}$.

Output : An allocation that approximately optimizes the NSW objective

Phase 1:

1 $\mathbf{x}_i^1 \leftarrow \emptyset$, $\forall i \in \mathcal{A}$ `// x_i^1's store the set of items allocated in Phase 1`
2 $\mathcal{G}^{rem} \leftarrow \mathcal{G}$ `// set of unallocated items before every iteration`
3 $t \leftarrow 0$ `// iteration counter`
4 **while** $\mathcal{G}^{rem} \neq \emptyset$ *and* $t \leq \lceil \log n \rceil$ **do**
5   Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with weights
      $\mathcal{W} = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(j)), \forall i \in \mathcal{A}, j \in \mathcal{G}^{rem}\}$
6   Compute a maximum weight matching $\mathcal{M}$ for Γ
7     $\mathbf{x}_i^1 \leftarrow \mathbf{x}_i^1 \cup \{j\}$, $\forall (i,j) \in \mathcal{M}$ `// allocate items to agents according to M`
8     $\mathcal{G}^{rem} \leftarrow \mathcal{G}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\}$; $t \leftarrow t + 1$ `// remove allocated items`
9 **end**

Phase 2:

10 For all i, $\mathbf{x}_i^2 \leftarrow \emptyset$ `// x_i^2's are the sets of items allocated in Phase 2`
11 **while** $\mathcal{G}^{rem} \neq \emptyset$ **do**
12   Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with weights
      $W = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(\mathbf{x}_i^2 \cup \{j\})), \forall i \in \mathcal{A}, j \in \mathcal{G}^{rem}\}$
13   Compute a maximum weight matching $\mathcal{M}$ for Γ
14     $\mathbf{x}_i^2 \leftarrow \mathbf{x}_i^2 \cup \{j\}$, $\forall (i,j) \in \mathcal{M}$ `// allocate items to agents according to M`
15     $\mathcal{G}^{rem} \leftarrow \mathcal{G}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\}$ `// remove allocated items`
16 **end**

Phase 3:

17 $\mathcal{G}^{rem} \leftarrow \bigcup_i \mathbf{x}_i^1$ `// release items allocated in Phase 1`
18 Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with
    $W = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(\mathbf{x}_i^2 \cup \{j\})), \forall i \in \mathcal{A}, j \in \mathcal{G}^{rem}\}$
19 Compute a maximum weight matching $\mathcal{M}$ for Γ
20 $\mathbf{x}_i^2 \leftarrow \mathbf{x}_i^2 \cup \{j\}$, $\forall (i,j) \in \mathcal{M}$ `// allocate items to agents according to M`
21 Arbitrarily allocate rest of the items to agents, let $\mathbf{x} = \{\mathbf{x}_i\}_{i \in \mathcal{A}}$ denote the final allocation
22 **return x**

---

agent i loses in t^{th} iteration is the intersection of goods allocated in t^{th} iteration with her optimal bundle. We refer to this set by $\mathcal{S}_i^t$, i.e., $\mathcal{S}_i^t := \mathcal{S}^t \cap \mathbf{x}_i^*$. Further, let the number of items in $\mathcal{S}_i^t$ be k_i^t.

For the analysis of RepReMatch, we also introduce the notion of *attainable items* for every iteration. We note that $\mathcal{S}_i^t$ is the set of an agent's preferred items that she lost to other agents. The items that are now left from the optimal bundle are referred to as the set of *attainable* items of the agent. Note that in any matching, every agent gets an item equivalent to her best attainable item, i.e., an item for which her marginal valuation (over her current allocation) is at least equal to that of her highest marginally valued attainable item.

For all i, we denote the intersection of the set of *attainable* items in the tth iteration and agent i's optimal bundle $\mathbf{x}_i^*$ by $\bar{\mathbf{x}}_{i,t}^*$, and let $u_i^* = v_i(\bar{\mathbf{x}}_{i,1}^*) = v_i(\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1)$ be the total valuation of *attainable* items at the first iteration of Phase 2. In the following lemma, we prove a lower bound on the marginal valuation of the set of *attainable* items over the set of items the algorithm already allocated to the agent. Intuitively, the total value remaining after t iterations is the difference between the total value that it is possible to attain and the value lost to other agents (second and third terms in (2)) and the value that has been allocation to this agent herself (last term in (2)). This easily follows in the case of additive valuations; we give proof for submodular valuations below.

LEMMA 3.2. *For any $j \in [\tau_i^2 - 1]$,*

$$v_i(\bar{\mathbf{x}}_{i,j+1}^* \mid h_i^1, \ldots, h_i^j) \geq u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^j) \ . \quad (2)$$

PROOF. We prove this lemma using induction on the number of iterations t. Consider the base case when $t = 2$. Agent i has already been allocated h_i^1. She now has at most $\bar{\tau}_i^* - k_i^1$ items left from $\bar{\mathbf{x}}_i^*$ that are not yet allocated. In the next iteration, the agent loses k_i^2 items to other agents and receives h_i^2. Each of the remaining $\bar{\tau}_i^* - k_i^1$ items have marginal utility at most $v_i(h_i^1)$ over $\emptyset$. Thus, the marginal utility of these items over h_i^1 is also at most $v_i(h_i^1)$. We bound the total marginal valuation of $\bar{\mathbf{x}}_{i,2}^*$ over $\{h_i^1\}$, by considering two cases.

Case 1: $h_i^1 \notin \bar{\mathbf{x}}_{i,1}^*$: By monotonicity of v_i, $v_i(\bar{\mathbf{x}}_{i,2}^* \mid h_i^1) \geq v_i(\bar{\mathbf{x}}_{i,2}^*) - v_i(h_i^1) = v_i(\bar{\mathbf{x}}_{i,1}^* \setminus \mathcal{S}_i^2) - v_i(h_i^1)$.

Case 2: $h_i^1 \in \bar{\mathbf{x}}_{i,1}^*$: Here, $v_i(\bar{\mathbf{x}}_{i,2}^* \mid h_i^1) = v_i(\bar{\mathbf{x}}_{i,2}^* \cup \{h_i^1\}) - v_i(h_i^1) = v_i(\bar{\mathbf{x}}_{i,1}^* \setminus \mathcal{S}_i^2) - v_i(h_i^1)$.
In both cases, submodularity of valuations and the fact that for all $j \in \mathcal{S}_i^2, v_i(j) \leq v_i(h_i^1)$ implies,

$$v_i(\bar{\mathbf{x}}_{i,2}^* \mid h_i^1) \ \geq \ v_i(\bar{\mathbf{x}}_{i,1}^*) - v_i(\mathcal{S}_i^2) - v_i(h_i^1) \ \geq \ u_i^* - k_i^2 v_i(h_i^1) - v_i(h_i^1),$$

proving the base case. Now assume the lemma is true for all $t \leq r$ iterations, for some r, i.e.,

$$v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^{r-1}) \geq u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^{r-1}).$$

Consider the $(r + 1)^{st}$ iteration. Again, we analyze two cases.
Case 1: $h_i^r \notin \bar{\mathbf{x}}_{i,r}^*$:
$$\begin{aligned}
v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) &= v_i(\bar{\mathbf{x}}_{i,r}^* \setminus \mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^r) \\
&\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^r) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^r) \\
&\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^r) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}) \\
&\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}).
\end{aligned}$$

The submodularity of v_i gives the first two inequalities, and the monotonicity of v_i implies the last.
Case 2: $h_i^r \in \bar{\mathbf{x}}_{i,r}^*$:
$$\begin{aligned}
v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) &= v_i(\bar{\mathbf{x}}_{i,r+1}^* \cup \{h_i^r\} \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) \\
&= v_i(\bar{\mathbf{x}}_{i,r}^* \setminus \mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) \\
&\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}).
\end{aligned}$$

Here, the second expression follows as $\bar{\mathbf{x}}_{i,r}^* = \bar{\mathbf{x}}_{i,r+1}^* \cup \{h_i^r\} \cup \mathcal{S}_i^{r+1}$, and the last follows from the definition of submodularity of the valuations.

In both cases, from the induction hypothesis, we get,

$$v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) \geq u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^{r-1})$$

$$- v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}).$$

Finally, since RepReMatch assigns the item with the highest marginal utility from the set of *attainable* items, and each item in $\mathcal{S}_i^{r+1}$ is attainable at r^{th} iteration,

$$v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) \geq u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^{r-1})$$

$$- v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) - k_i^{r+1} v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1})$$

$$= u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^r). \qquad \square$$

The above lemma directly allows us to give a lower bound on the marginal valuation of the item received by the agent in $(j+1)^{th}$ iteration over the items received in previous iterations. We state and prove this in the following corollary.

COROLLARY 3.3. *For any $j \in [\tau_i^2 - 1]$,*

$$v_i(h_i^{j+1} \mid h_i^1, \ldots, h_i^j) \geq \frac{1}{\bar{\tau}_i^* - \sum_{t=1}^{j+1} k_i^t} \left(u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, \ldots, h_i^j) \right).$$

PROOF. In any setting with a set of items $\mathcal{S} = \{s_1, \ldots s_k\}$, and a monotone submodular valuation v on this set, if $v(\mathcal{S}) = u$, then there exists an item $s \in \mathcal{S}$ such that $v(s) \geq u/k$. Thus, with $\mathcal{S} = \bar{\mathbf{x}}_{i,j+1}^*$, $k = \bar{\tau}_i^* - \sum_{t=1}^{j+1} k_i^t$, for the submodular valuation function $v_i(\cdot \mid \{h_i^1, \ldots, h_i^j\})$, we can say that at iteration $j + 1$, h_i^{j+1} will have a marginal valuation at least,

$$\frac{1}{\bar{\tau}_i^* - \sum_{t=1}^{j+1} k_i^t} v_i(\bar{\mathbf{x}}_{i,j+1}^* \mid h_i^1, \ldots, h_i^j) .$$

Together with Lemma 3.2, this proves the corollary. Note that at any iteration t, if the received item h_i^t is from $\bar{\mathbf{x}}_{i,t}^*$, then the denominator reduces further by 1, and the bound still holds. $\qquad \square$

In the following lemma, we give a lower bound on the total valuation of the items received by the agent in Phase 2.

LEMMA 3.4. $v_i(h_i^1, \ldots, h_i^{\tau_i^2}) \geq \frac{u_i^*}{n}.$

PROOF. Recall that u_i^* is the valuation of the items from $\bar{\mathbf{x}}_i^*$ *after* she loses items in $\mathcal{S}_i^1$ to other agents in the first iteration of Phase 2 and $\bar{\tau}_i^*$ is the number of items in $\bar{\mathbf{x}}_i^*$. From Corollary 3.3, total valuation of the items obtained by agent i in Phase 2 is bounded as follows.

$$v_i(h_i^1, \ldots h_i^{\tau_i^2}) = v_i(h_i^1, \ldots, h_i^{\tau_i^2-1}) + v_i(h_i^{\tau_i^2} \mid h_i^1, \ldots, h_i^{\tau_i^2-1})$$

$$\geq v_i(h_i^1, \ldots, h_i^{\tau_i^2-1}) + \frac{1}{\bar{\tau}_i^* - \sum_{t=0}^{\tau_i^2-1} k_i^{t+1}} \left(u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{\tau_i^2-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots h_i^{t-1}) \right.$$

$$\left. - v_i(h_i^1, \ldots, h_i^{\tau_i^2-1}) \right).$$

By definition, τ_i^2 is the last iteration of Phase 2 in which agent i gets matched to some item. After this iteration, at most n items from her optimal bundle remain unallocated, else she would have received one more item in the $(\tau_i^2 + 1)^{st}$ iteration. This means the optimal number of items $\bar{\tau}_i^* - \sum_{t=0}^{\tau_i^2-1} k_i^{t+1} \le n$, hence the denominator of the second term in the above equation is at most n. Again, we note here that if at any iteration t, the item assigned to agent i was from $\bar{\mathbf{x}}_{i,t}^*$, then the denominator will be further reduced by 1 for all such iterations, and the inequality still remains true when k_i^t is replaced by $k_i^t + 1$. Combined with the fact that an agent can lose at most $n - 1$ items in every iteration, we get $k_i^t \le n - 1$, implying,

$$v_i(h_i^1, \ldots, h_i^{\tau_i^2})$$

$$\ge v_i(h_i^1, \ldots, h_i^{\tau_i^2-1}) + \frac{1}{n}\left(u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{\tau_i^2-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots h_i^{t-1}) - v_i(h_i^1, \ldots, h_i^{\tau_i^2-1})\right)$$

$$\ge v_i(h_i^1, \ldots, h_i^{\tau_i^2-1}) + \frac{1}{n}\left(u_i^* - (n-1) v_i(h_i^1) - \sum_{t=2}^{\tau_i^2-1} (n-1) v_i(h_i^t \mid h_i^1, \ldots h_i^{t-1}) - v_i(h_i^1, \ldots, h_i^{\tau_i^2-1})\right)$$

$$= v_i(h_i^1, \ldots, h_i^{\tau_i^2-1}) + \frac{1}{n}\left(u_i^* - (n-1) v_i(h_i^1, h_i^2 \ldots, h_i^{\tau_i^2-1}) - v_i(h_i^1, h_i^2 \ldots, h_i^{\tau_i^2-1})\right) = \frac{u_i^*}{n} \ . \qquad \square$$

REMARK 3.1. *In Lemma 3.2 and its subsequent Corollary 3.3 and Lemma 3.4, if $u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) - v_i(h_i^1, \ldots, h_i^j)$ becomes negative for any $j \in [\tau_i^2 - 1]$, then we have*

$$u_i^* \le k_i^2 v_i(h_i^1) + \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) + v_i(h_i^1, \ldots, h_i^j)$$

$$\le (n-1) v_i(h_i^1) + \sum_{t=2}^{j} (n-1) v_i(h_i^t \mid h_i^1, \ldots, h_i^{t-1}) + v_i(h_i^1, \ldots, h_i^j)$$

$$= n \cdot v_i(h_1, \ldots, h_i^j) \le n \cdot v_i(h_1, \ldots, h_i^{\tau_i^2-1}),$$

which implies that Lemma 3.4 holds.

We now bound the minimum valuation that can be obtained by every agent in Phase 3. Recall that g_i^1 is the item that gives the highest marginal utility over the empty set to agent i. Before proceeding, we define

$$\mathcal{G}_i^1 := \{g \in \mathcal{G} \mid v_i(g \mid \emptyset) \ge v_i(g_i^1 \mid \emptyset)\} \ .$$

LEMMA 3.5. *Consider the complete bipartite graph where the set of agents $\mathcal{A}$, and the set of items allocated in the first Phase of* RepReMatch *are the parts, and edge weights are the weighted logarithm of the agent's valuation for the bundle of items containing the item adjacent to the edge and items allocated in Phase 2. That is, consider $\Gamma(\mathcal{A}, \mathcal{G} = \bigcup_i \mathbf{x}_i^1, \mathcal{W} = \{w(i,j) = \eta_i \log(v_i(\{j\} \cup \mathbf{x}_i^2))\})$. In this graph, there exists a matching where each agent i gets matched to an item from their highest valued set of items $\mathcal{G}_i^1$.*

PROOF. Among all feasible matchings between the set of agents and the set of items, say T, allocated after t iterations of Phase 1, consider the set of matchings $\mathcal{M}$ where each agent i whose $\mathcal{G}_i^1 \subseteq T$ is matched to some item in $\mathcal{G}_i^1$. Arbitrarily pick a matching from $\mathcal{M}$ where a maximum number of agents are matched to an item from their $\mathcal{G}_i^1$s. Denote this matching by $\mathcal{M}^t$. Since

$|\bigcup_{i \in S} \mathcal{G}_i^1| \geq |\mathcal{S}|$ for every set S of agents, in $\mathcal{M}^t$, each agent i, who is not matched to an item from their $\mathcal{G}_i^1$, has at least one item of $\mathcal{G}_i^1$ still unallocated after t iterations.

Let $\mathcal{A}_t$ denote the set of agents that are not matched to any item from their $\mathcal{G}_i^1$ in $\mathcal{M}^t$. We prove by induction on t that $|\mathcal{A}_t| \leq n/2^t$.

For the base case, when $t = 1$, we count the number of agents who did not receive any item from their own $\mathcal{G}_i^1$ in the maximum weight matching of the algorithm. We know that before the first iteration, every item is unallocated. An agent will not receive any item from $\mathcal{G}_i^1$ only if all items from this set are allocated to other agents in the matching. Hence, if α agents did not receive any item from their $\mathcal{G}_i^1$, all items from at least α number of $\mathcal{G}_i^1$ sets got matched to some agent(s) in the first matching. If $\alpha < n/2$, then more than $n/2$ agents themselves received some item from their $\mathcal{G}_i^1$. If $\alpha \geq n/2$, then at least α items, each from a different $\mathcal{G}_i^1$ were allocated. In either case, releasing the allocation of the first matching releases at least $n/2$ items, each belonging to a distinct agent's $\mathcal{G}_i^1$. Hence, in $\mathcal{M}^1$ at least $n/2$ agents receive an item from their $\mathcal{G}_i^1$, and $|\mathcal{A}_1| \leq n/2$.

For the inductive step, we assume the claim is true for the first t iterations. That is, for every $k \leq t$, in $\mathcal{M}^k$, at most $n/2^k$ agents do not receive an item from their $\mathcal{G}_i^1$'s.

Before the $(t+1)^{st}$ iteration begins, we know that for every agent i in $\mathcal{A}_t$, at least one item from their $\mathcal{G}_i^1$ is still unallocated. Again by the reasoning of the base case, at least half of the agents in $\mathcal{A}_t$ will have some item from their $\mathcal{G}_i^1$ allocated in the $(t+1)^{st}$ matching (possibly to some other agent). Hence, in $\mathcal{M}^{(t+1)}$, $|\mathcal{A}_{(t+1)}| \leq |\mathcal{A}_t|/2$. By the inductive hypothesis, $|\mathcal{A}_{(t+1)}| \leq n/2^{(t+1)}$.   □

We now restate and prove Theorem 3.1.

THEOREM 3.1. *The* NSW *objective of allocation* $\mathbf{x}$, *output by* RepReMatch *for asymmetric submodular* NSW *problem, is at least* $1/2n(\log n + 2)$ *times the optimal* NSW, *denoted as* OPT, *i.e.,* $\mathrm{NSW}(\mathbf{x}) \geq \frac{1}{2n(\log n + 2)} \mathrm{OPT}$.

PROOF. From Lemma 3.4,

$$v_i(h_i^1, \ldots, h_i^{\tau_i^2}) \geq \frac{u_i^*}{n}.$$

By Lemma 3.5, giving each agent her own g_i^1 or some item, denoted by say h_i^{1*}, that gives her a marginal utility over $\emptyset$ at least as much as $v_i(g_i^1)$ is a feasible matching before Phase 3 begins. Therefore, we get,

$$\mathrm{NSW}(\mathbf{x}) \geq \left(\prod_{i=1}^n (v_i(h_i^{1*}, h_i^2, \ldots, h_i^{\tau_i^2}))^{\eta_i} \right)^{1/(\sum_{i=1}^n \eta_i)}. \tag{3}$$

Since the valuation functions are monotonic,

$$v_i(h_i^{1*}, h_i^2, \ldots, h_i^{\tau_i^2}) \geq v_i(h_i^{1*}) \geq v_i(g_i^1).$$

Phase 1 of the algorithm runs for $\lceil \log n \rceil$ iterations and each iteration allocates n items. Thus, $|\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*| \leq n \lceil \log n \rceil$ and $|\mathcal{S}_i^1| \leq n$ implying, $|(\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1| \leq n(\log n + 2)$. Thus,

$$v_i(g_i^1) \geq \frac{1}{n(\log n + 2)} v_i((\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1).$$

Also,

$$v_i(h_i^{1*}, h_i^1, \ldots, h_i^{\tau_i^2}) \geq v_i(h_i^1, \ldots, h_i^{\tau_i^2}) \geq \frac{u_i^*}{n} = \frac{1}{n} v_i(\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1).$$

Thus,

$$v_i(h_i^{1*}, h_i^1, \ldots, h_i^{\tau_i^2}) \geq \frac{1}{2} \left(\frac{1}{n(\log n + 2)} v_i((\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1) + \frac{1}{n} v_i(\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1) \right)$$

$$\geq \frac{1}{2} \frac{1}{n(\log n + 2)} v_i(((\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*) \cup \mathcal{S}_i^1) \cup (\bar{\mathbf{x}}_i^* \setminus \mathcal{S}_i^1))$$

$$= \frac{1}{2} \frac{1}{n(\log n + 2)} v_i(\mathbf{x}_i^*) \ .$$

The second inequality follows from the submodularity of valuations. The last bound, together with (3) gives,

$$\text{NSW}(\mathbf{x}) \ \geq \ \left(\prod_{i=1}^{n} \left(\frac{1}{2} \frac{1}{n(\log n + 2)} v_i(\mathbf{x}_i^*) \right)^{\eta_i} \right)^{1/\sum_i \eta_i} \ \geq \ \frac{1}{2} \frac{1}{n(\log n + 2)} \text{OPT} \ . \qquad \square$$

REMARK 3.2. *We remark that even if Phases 1 and 2 perform some kind of repeated matchings, the edge weight definitions make them different. In the proof of Lemma 3.5, we require that a maximum weight matching matches agents to items according to agent valuations for the single item. That is, in all iterations of Phase 1, the edge weights of the graph in the future are the valuation of the agent for the set containing the single item, and not the increase in the agent's valuation upon adding this item to her current allocation. These quantities are different when the valuations are submodular. For lower bounding the valuation from the lower ranked items, we need to consider the marginal increase, as defined in Phase 2. However, Lemma 3.5 may not hold true if the marginal increase in valuations is considered for the initial iterations, hence Phase 1 is required.*

4 HARDNESS OF APPROXIMATION

We complement our results for the submodular case with a $\frac{e}{(e-1)}$-factor hardness of approximation. Formally, we prove the following theorem.

THEOREM 4.1. *Unless* P = NP, *there is no polynomial time $\frac{e}{(e-1)}$-factor approximation algorithm for the submodular NSW problem, even when agents are symmetric and have identical valuations.*

PROOF. We show this using the hardness of approximation result of the ALLOCATION problem proved in [30]. We first summarize the relevant parts of [30]. The ALLOCATION problem is to find an allocation of a set of indivisible items among a set of agents with monotone submodular utilities for the items, such that the sum of the utilities of all agents is maximized. Note that if the valuation functions were additive, the problem is trivial, and an optimal allocation gives every item to the agent who values it the most. To obtain a hardness of approximation result for the submodular case, the MAX-3-COLORING problem is reduced to the ALLOCATION problem. MAX-3-COLORING, the problem of determining what fraction of edges of a graph can be properly colored when 3 colors are used to color all vertices of the graph, is known to be NP-Hard to approximate within some constant factor c. The reduction from MAX-3-COLORING generates an instance of ALLOCATION with symmetric agents having identical submodular valuation functions for the items. The reduction is such that for instances of MAX-3-COLORING with optimal value 1, the corresponding ALLOCATION instance has an optimal value of nV, where n is the number of agents in the instance, and V is a function of the input parameters of MAX-3-COLORING. In this case, every agent receives a set of items of utility V. For instances of MAX-3-COLORING with optimal value at most c, it is shown that the optimal sum of utilities of the resulting ALLOCATION instance cannot be higher than $(1 - 1/e)nV$.

For proving the hardness of the submodular NSW problem, observe that the input of the ALLOCATION and NSW problems are the same. So, we consider the instance generated by the reduction as that of an NSW maximizing problem. We can prove the following claims from the results of [30].

- If the optimal value of MAX-3-COLORING is 1, then the NSW of the reduced instance is V. As every agent receives a set of items of value V, the NSW is also V.
- If the optimal value of MAX-3-COLORING is at most c, then the NSW is at most $(1 - 1/e)V$. Applying the AM-GM inequality establishes that the NSW is at most $1/n$ times the sum of utilities, which is proven to be at most $(1 - 1/e)nV$.

As MAX-3-COLORING cannot be approximated within a factor c, thus NSW of a problem with submodular utilities cannot be approximated within a factor $\frac{e}{(e-1)}$.

As the ALLOCATION problem, now considered as an NSW problem, had symmetric agents and identical submodular valuation functions, the NSW problem also satisfies these properties.   □

5 SPECIAL CASES

5.1 Submodular NSW with Constant Number of Agents

In this section, we describe a constant factor algorithm for a special case of the submodular NSW problem. Specifically, we prove the following theorem.

THEOREM 5.1. *For any constant $\epsilon > 0$ and a constant number of agents $n \geq 2$, there is a $(1 - 1/e - \epsilon)$-factor approximation algorithm for the NSW problem with monotone submodular valuations, in the value oracle model. Additionally, this is the best possible factor independent of n, and any factor better than $(1 - (1 - 1/n)^n + \epsilon)$ would require exponentially many queries unless $P = NP$.*

The key results that establish this theorem are from the theory of submodular function maximization developed in [14]. The broad approach for approximately maximizing a discrete monotone submodular function is to optimize a popular continuous relaxation of the same, called the multilinear extension, and round the solution using a randomized rounding scheme. We will use an algorithm that approximately maximizes multiple discrete submodular functions, described in [14], as the main subroutine of our algorithm for the submodular NSW problem. Hence first, we give an overview, starting with a definition of the multilinear extension.

Definition 5.2 (Multilinear Extension of a submodular function). Given a discrete submodular function $f : 2^m \to \mathbb{R}_+$, its multilinear extension $F : [0,1]^m \to \mathbb{R}_+$, at a point $y \in [0,1]^m$, is defined as the expected value of $f(z)$ at a point $z \in \{0,1\}^m$ obtained by rounding y such that each coordinate y_i is rounded to 1 with probability y_i, and to 0 otherwise. That is,

$$F(y) = \mathbb{E}[f(z)] = \sum_{X \subseteq [m]} f(X) \prod_{i \in X} y_i \prod_{i \notin X} (1 - y_i).$$

The following theorem proves that the multilinear extensions of multiple discrete submodular functions defined over a matroid polytope can be simultaneously approximated to optimal values within constant factors.

THEOREM 5.3. *[14] Consider monotone submodular functions $f_1, \ldots, f_n : 2^N \to \mathbb{R}_+$, their multilinear extensions $F_i : [0,1]^N \to \mathbb{R}_+$ and a matroid polytope $P \subseteq [0,1]^N$. There is a polynomial time algorithm which, given $V_1, ..., V_n \in \mathbb{R}_+$, either finds a point $x \in P$ such that $F_i(x) \geq (1 - 1/e)V_i$ for each i, or returns a certificate that there is no point $x \in P$ such that $F_i(x) \geq V_i$ for all i.*

Given a discrete monotone submodular function f defined over a matroid, a rounding scheme called the *swap rounding* algorithm can be applied to round a solution of its multilinear extension to a feasible point in the domain of f, which is an independent set of the matroid. At a high level, in the rounding scheme, it is first shown that every solution of the multilinear extension can be expressed as a convex combination of independent sets such that for any two sets S_0 and S_1 in the convex combination, there is at least one element in each set that is not present in the other,

---

Algorithm 3: Submodular NSW with constant number of asymmetric agents

---

Input : A set $\mathcal{A}$ of n agents with weights η_i, $\forall i \in \mathcal{A}$, a set $\mathcal{G}$ of m indivisible items, and monotone submodular valuations $u_i : 2^{\mathcal{G}} \to \mathbb{R}_+$.

Output : An allocation that approximates the NSW.

---

1 Define $\delta > 0$ as any small positive constant
2 Define for all $i \in \mathcal{A}$: $U_i \leftarrow u_i(\mathcal{G})$, $L_i \leftarrow \min_{g \in Goods} u_i(g)$,
   $R_i \leftarrow \{v = L_i * (1 + \delta)^{k_i} \mid v \leq U_i, \ k_i \text{ is a non-negative integer}\}$ // Range of possible
   values of the Nash optimal bundle of each agent
3 Initialize $max = 0$ // optimal Nash welfare value
4 **for** all vectors $V = [V_1, V_2, \ldots, V_n]$ in the set $\mathsf{V}^s := \{V \mid \forall i \in [n] : V_i \in R_i\}$ **do**
5 | **if** there is an allocation $\mathbf{x}$ of $\mathcal{G}$ such that $u_i(\mathbf{x}_i) \geq (1 - 1/e)V_i$ for all i **then**
6 | | **if** $\sum_{i \in A} \eta_i * \log(u_i(\mathbf{x}_i)) \geq max$ **then**
7 | | | $\mathbf{x}^* \leftarrow \mathbf{x}, max \leftarrow \sum_{i \in A} \eta_i * \log(u_i(\mathbf{x}_i))$ // store optimal NSW allocation
8 | | **end**
9 | **end**
10 **end**
11 **return** $\mathbf{x}^*$

---

that is $\exists e_0 \in S_0 \backslash S_1$ and $\exists e_1 \in S_1 \backslash S_0$. The rounding method then iteratively merges two arbitrarily chosen sets S_0 and S_1 into one new set as follows. Until both sets are not the same, one set S_i is randomly chosen with probability proportional to the coefficient of its original version in the convex combination β_i, that is, S_i is chosen with probability $\beta_i/(\beta_0 + \beta_1)$, and altered by removing e_i from it and adding e_{1-i}. The coefficient of the new set obtained by this merge process is the sum of those of the sets merged, i.e., $\beta_0 + \beta_1$.

The following lower tail bound proves that with high probability, the loss in the function value by swap rounding is not too much.

THEOREM 5.4. *[14] Let $f : \{0, 1\}^n \to \mathbb{R}_+$ be a monotone submodular function with marginal values in $[0, 1]$, and $F : [0, 1]^n \to \mathbb{R}_+$ its multilinear extension. Let $(x_1, ..., x_n) \in P(M)$ be a point in a matroid polytope and $(X_1, ..., X_n) \in \{0, 1\}^n$ a random solution obtained from it by randomized swap rounding. Let $\mu_0 = F(x_1, ..., x_n)$ and $\delta > 0$. Then*

$$Pr[f(X_1, ..., X_n) \leq (1 - \delta)\mu_0] \leq e^{-\mu_0 \delta^2 / 8}.$$

In short, for a matroid $M(X, I)$, given monotone submodular functions $f_i : \{0, 1\}^m \to \mathbb{R}_+, i \in [n]$ over the matroid polytope, and values $v_i, i \in [n]$, there is an efficient algorithm that determines if there is an independent set $S \in I$ such that $f_i(S) \geq (1 - 1/e)v_i$ for every i.

To use this algorithm to solve the submodular NSW problem, we define a matroid $M(X, I)$ as follows. This construction was first described in [34] and used to approximate the submodular welfare in [43]. From the sets of agents $\mathcal{A}$ and items $\mathcal{G}$, we define the ground set $X = \mathcal{A} \times \mathcal{G}$. The independent sets are all feasible integral allocations $I = \{S \subseteq X \mid \forall j : |S \cap \{\mathcal{A} \times \{j\}\}| \leq 1\}$. The valuation function of agent i, $u_i : \{0, 1\}^m \to \mathbb{R}_+$ translates naturally to submodular function over this matroid $f_i : I \to \mathbb{R}_+$, with $f_i(S) = u_i(\mathcal{G}_i)$, where $\mathcal{G}_i = \{j \in \mathcal{G} \mid (i, j) \in S\}$. With this construction, for any set of values $V_i, i \in [n]$, checking if there is an integral allocation of items that give valuations at least (approximately) V_i to each agent i is equivalent to checking if there is an independent set in this matroid that has value V_i for every agent i.

The algorithm for approximating the NSW is now straightforward and given in Algorithm 3. Essentially, we enumerate all possible values each agent can receive in their Nash optimal bundle, rounded to the nearest term in some geometric progression. Using this, we enumerate all the possible vectors of (rounded) utility values received by all agents in the Nash optimal allocation. For each vector $V = \{V_1, V_2 \ldots, V_n\}$, we check if there is an allocation of $\mathcal{G}$ where each agent receives a bundle of utility at least $(1 - 1/e)V_i$, and return the allocation with the highest Nash welfare value among these.

We first show the (weakly) polynomial run time of this process.

LEMMA 5.5. *Algorithm 3 ends in time polynomial in n, m and $\log_{(1+\delta)}(U_i/V_i)$.*

PROOF. First, note that the utility value of the bundle received by any agent i in their Nash optimal allocation, or more generally in any allocation, is between $L_i = \min_{g \in \mathcal{G}} u_i(g)$ and $U_i = u_i(\mathcal{G})$. The set R_i stores all values in this range that are of the form $L_i * (1 + \delta)^{k_i}$, for some positive constant δ and a non-negative integer k_i. The number of values in R_i is $|R_i| \leq \log_{(1+\delta)}(U_i/L_i) = \log_{(1+\delta)}(U_i) - \log_{(1+\delta)}(L_i)$. As both the values in this difference are polynomials, $|R_i|$ is a polynomial.

The number of utility vectors enumerated, and therefore the number of iterations of the for loop, is bounded as $|V^s| \leq (\max_i |R_i|)^n$. As there are constantly many agents, this is a polynomial. As each iteration runs in polynomial time, the overall run time is polynomial in n, m and $\log(U_i/V_i)$.   □

The hardness claim in Theorem 5.1 follows from the proof of Theorem 4.1. It was shown that in the case where the optimal value of the MAX-3-COLORING instance was 1, every agent in the reduced NSW instance received a bundle of items of value V, else the total NSW could not be more than $(1 - (1 - 1/n)^n)V$.

5.2 Symmetric Additive NSW

We now prove that SMatch gives an allocation that also satisfies the EF1 property, making it not only approximately efficient but also a fair allocation. EF1 is formally defined as follows.

Definition 5.6 ([10]). Envy-Free up to one item (EF1): An allocation $\mathbf{x}$ of m indivisible items among n agents satisfies the envy-free up to one item property, if for any pair of agents $i, \hat{i}$, either $v_i(\mathbf{x}_i) \geq v_i(\mathbf{x}_{\hat{i}})$, or there exists some item $g \in \mathbf{x}_{\hat{i}}$ such that $v_i(\mathbf{x}_i) \geq v_i(\mathbf{x}_{\hat{i}} \setminus \{g\})$.

That is, if an agent i values another agent $\hat{i}$'s allocation more than her own, which is termed commonly by saying agent i *envies* agent $\hat{i}$, then there must be some item in $\hat{i}$'s allocation upon whose removal this envy is eliminated.

THEOREM 5.7. *The output of SMatch satisfies the EF1 fairness property.*

PROOF. For every agent i and $j \geq 1$, the item h_j^i allocated to i in the jth iteration of SMatch is valued more by i than all items $h_k^{i'}$, $k > j$ allocated to any other agent i' in the future iterations, as otherwise i would have been matched to the other higher valued item in the jth matching. Hence, $\sum_{t=1}^{j} v_i(h_t^i) \geq \sum_{t=2}^{j} v_i(h_t^{i'})$. That is, after removing the first item $h_1^{i'}$ from any agent's bundle, the sum of valuations of the remaining items for agent i is not higher than her current total valuation. Thus, after removing the item allocated to any agent in the first matching, agent i does not envy the remaining bundle, making the allocation EF1.   □

REMARK 5.1. *The same proof implies that our algorithm satisfies the strong EF1 property, defined in [18]. Intuitively, an allocation satisfies the strong EF1 property if, upon removing the same item from agent i bundle, no other agent j envies i for all i and j. Formally,*

Definition 5.8 (Strong EF1). An allocation $\mathbf{x}$ satisfies strong EF1 if for every agent $i \in \mathcal{A}$, there exists an item $g_i \in \mathbf{x}_i$ such that no other agent envies the set $\mathbf{x}_i \backslash \{g_i\}$, i.e., $\forall j \in \mathcal{A}, \ v_j(\mathbf{x}_j) \geq v_j(\mathbf{x}_i \backslash \{g_i\})$.

5.3 Symmetric Restricted Additive NSW

For the special case when the valuations are restricted, meaning the valuation of any item v_{ij} is either some value v_j or 0, we now prove SMatch gives a constant factor approximation to the optimal NSW.

THEOREM 5.9. *SMatch solves the symmetric NSW problem for restricted additive valuations within a factor* 1.45 *of the optimal.*

PROOF. We prove that x^*, the allocation returned by SMatch, is Pareto Optimal (PO). Combined with the statement of Theorem 5.7, and a result of [8], which proves that any allocation that satisfies both EF1 and PO approximates NSW with symmetric, additive valuations within a 1.45 factor, we get the required result. An allocation of items x is called Pareto Optimal when there is no other allocation x' where every agent gets at least as much utility as in x, and at least one agent gets higher utility. In the case of the restricted valuation, every item adds a valuation of either 0 or v_j to some agent's utility. Thus, the sum of valuations of all agents in any allocation is at most $\sum_j v_j$. Observe that SMatch can easily be modified to allocate every item to some agent with a non-zero valuation. Then, the sum of valuations of all agents in the allocation returned by SMatch is $\sum_j v_j$. No other allocation can give an agent strictly higher utility without decreasing another agent's utility. Hence, x^* is a Pareto Optimal allocation. □

REMARK 5.2. *We remark that Theorem 5.7 also holds for general additive valuations. However, the PO property does not always hold for the general case. Consider for example the case where we have two agents* $\{A, B\}$ *and four items* $\{g_1, g_2, g_3, g_4\}$. *Agent* A *values the items at* $\{2 + \epsilon, 2, \epsilon, \epsilon\}$ *and agent* B *values them at* $\{1, 1, 1, 1\}$. *SMatch allocates items* g_1, g_3 *to agent* A *and items* g_2, g_4 *to agent* B. *However, we can swap items* g_2 *and* g_3 *to get an allocation that Pareto dominates the allocation output by the algorithm.*

6 TIGHTNESS OF THE ANALYSIS

6.1 Subadditive Valuations

The matching approach does not extend to agents with subadditive valuation functions. Here the valuation functions satisfy the subadditivity property:

$$v(\mathcal{S}_1 \cup \mathcal{S}_2) \leq v(\mathcal{S}_1) + v(\mathcal{S}_2),$$

for any subsets $\mathcal{S}_1, \mathcal{S}_2$ of the set of items $\mathcal{G}$.

A counter-example that exhibits the shortcomings of the approach is as follows. Consider an instance with 2 agents and m items. Assume m is even. Denote the set of items by $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$. Let $\mathcal{G}_1 = \{g_1, g_2, \ldots, g_{m/2}\}$ and $\mathcal{G}_2 = \{g_{m/2+1}, \ldots, g_m\}$. The valuation function for agent $i \in \{1, 2\}$ is as follows.

$$v_i(\mathcal{S}) = v_i(\mathcal{S}_1 \cup \mathcal{S}_2) = \max\{M, |\mathcal{S}_i| \cdot M\} \quad \forall \mathcal{S} \subseteq \mathcal{G}, \ \mathcal{S}_1 \subseteq \mathcal{G}_1, \ \mathcal{S}_2 \subseteq \mathcal{G}_2.$$

Note that these valuation functions are subadditive but not submodular.

The allocation that maximizes the NSW allocates $\mathcal{G}_1$ to agent 1 and $\mathcal{G}_2$ to agent 2. The optimal NSW is $mM/2$.

Now, RepReMatch may proceed in the following way. Since the marginal utility of each item over $\emptyset$ is M, the algorithm can pick any of the items for either of the agents. Suppose the algorithm

gives $g_{m/2+1}$ to agent 1 and g_1 to agent 2. In the next iteration, for agent 1 (2) the marginal utility of any item over $g_{m/2+1}$ (g_1) is 0. Thus, again, the algorithm is at liberty to allocate any item to either of the agents. Again, the algorithm gives exactly the opposite allocation compared to the optimal allocation and gives agent 1 item $g_{m/2+2}$ and gives agent 2 item g_2. For each iteration, this process repeats, and ultimately the bundles allocated by the algorithm are exactly opposite of those allocated by optimal. The re-matching step first releases $\lceil \log n \rceil$ matchings, or 2 items from both agent allocations, and re-matches them. This may not change the allocations as both agents have already received their best item. The NSW of the algorithm's allocation is $(M^2)^{1/2} = M$, giving an approximation ratio of $\Omega(m)$ with the optimal NSW. Even if we increase the number of agents, the factor cannot be independent of m, the number of items.

The problem in the subadditive case is the myopic nature of each iteration in RepReMatch. In each iteration, the algorithm only sees one step ahead. At any of the iterations, had the algorithm been allowed to pick and allocate multiple items instead of 1, it would have been able to select a subset of items from its correct optimal bundle.

This problem does not arise in the additive case because the valuation of an item here is independent of other items. Submodular valuations allow a minimum marginal utility over an agent's current allocation for items allocated in future iterations, hence this issue does not arise there too.

6.2 XOS Valuations

The following example shows that RepReMatch does not extend to XOS valuations either. XOS is a class of valuation functions that falls between subadditive and submodular, defined as follows. A set of additive valuation functions, say $\{\ell_1, \ldots, \ell_k\}$, is given, and the XOS valuation of a set of items $\mathcal{S}$ is the maximum valuation of this set according to any of these additive valuations. i.e., $v(\mathcal{S}) = \max_{i \in [k]} \{\ell_i(\mathcal{S})\}$.

To see why the algorithm does not extend to this class of functions, consider the following counterexample. We have $n = 2$ agents and $m = 2k$ items for some $k > 3$. Each agent $i \in \{1, 2\}$ has 2 valuation functions ℓ_1^i, ℓ_2^i. The following two tables pictorially depict these valuations. Each entry (ℓ_h^i, g_j), $h \in [2], i \in [2], j \in [2k]$ denotes agent i's valuation according to function ℓ_h^i for item g_j.

For agent 1:

	g_1	g_2	$\ldots$	g_k	g_{k+1}	g_{k+2}	g_{k+3}	g_{k+4}	$\ldots$	g_{2k}
ℓ_1^1	M	M	$\ldots$	M	0	0	0	0	$\ldots$	0
ℓ_2^1	0	0	$\ldots$	0	$M+\epsilon$	$M+\epsilon$	$M+\epsilon$	ϵ	$\ldots$	ϵ

For agent 2:

	g_1	g_2	g_3	g_4	$\ldots$	g_k	g_{k+1}	g_{k+2}	$\ldots$	g_{2k}
ℓ_1^2	0	0	0	0	$\ldots$	0	M	M	$\ldots$	M
ℓ_2^2	M+ϵ	M+ϵ	M+ϵ	ϵ	$\ldots$	ϵ	0	0	$\ldots$	0

Here, M is any large value, and $\epsilon > 0$ is negligible.

The allocation optimizing NSW clearly allocates the first k items to agent 1 and the next k items to agent 2, resulting in the NSW value Mk. RepReMatch on the other hand allocates items g_1, g_2 to agent 2 and items g_{k+1}, g_{k+2} to agent 1 in Phase 1. In Phase 2, it gives g_3 to agent 2 and g_{k+3} to agent 1. After this, for all other iterations of Phase 2, items $g_j, j \leq k$ have zero marginal utility for agent 1 and items $g_j, j \geq (k+1)$ have zero marginal utility for agent 2. Thus, RepReMatch allocates items $g_3 \ldots g_k$ to agent 2 and items $g_{k+3}, \ldots, g_{2k}$ to agent 1 in Phase 2. Phase 3 reallocates items of Phase $1 - g_1, g_2, g_{k+1}, g_{k+2}$ allocating g_{k+1}, g_{k+2} to agent 1 and g_1, g_2 to agent 2. Thus the NSW of the allocation given by RepReMatch is $(3M + k \cdot \epsilon)$. Hence, the approximation ratio of RepReMatch cannot be better than $(MK)/(3M + k\epsilon)$ or $\Omega(k) = \Omega(m)$ when the valuation functions are XOS.

6.3 Asymmetric Additive NSW

We describe an example to prove the analysis of this case is tight. Consider an NSW instance with n agents, referred by $\{1, 2 \ldots, n\}$ and m sets of n^2 items, referred by $\mathcal{G} = \{\mathcal{G}_i \mid i \in [m]\}$, where every $\mathcal{G}_i = \{g_{i,1} \ldots, g_{i,n^2}\}\}$. The first agent has weight W, while the remaining agents have weight 1. The valuation function of agent 1 is as follows.

$$v_1(g_{i,j}) = \begin{cases} M & j \in [n], i \in [m] \\ 0 & otherwise. \end{cases}$$

The remaining agents have valuations for items as follows.

$$\forall k \in [n], k \neq 1 : v_k(g_{i,j}) = \begin{cases} M + \epsilon & j \in [n], i \in [m], \epsilon > 0 \\ M + \bar{\epsilon} & (k-1)n + 1 \leq j \leq kn, i \in [m], \epsilon > \bar{\epsilon} > 0 \\ 0 & otherwise. \end{cases}$$

It is easy to verify that the optimal allocation that maximizes NSW gives all items $g_{i,j}$ for i between $(k-1)n + 1$ and kn to agent k. That is, the kth agent receives the kth set of n items from each of the m sets of n^2 items. SMatch, on the other hand, allocates items as follows. For the first graph, it computes u_i as $M(m-2)n$ for the first agent, and $(M + \epsilon)(m-2)n + (M + \epsilon')(mn)$ for a small $\epsilon' > 0$, for the rest. The first max weight matching then allocates to each agent one item from agent 1's optimal bundle. The following iterations also err as follows. Until the first n items from every set are allocated, irrespective of the agent weights, every agent receives one item from this set if available. In the remaining matchings, agent 1 does not receive any item, and the other agents get all items in their optimal bundles. The ratio of the NSW products of the optimal allocation and the algorithm's allocation is as follows.

$$
\begin{aligned}
\frac{\text{NSW}(\mathbf{x})}{\text{OPT}} &\leq \left(\frac{(mM)^W (mn \cdot (M + \bar{\epsilon}) + m(M + \epsilon))^{n-1}}{(mn \cdot M)^W (mn \cdot (M + \bar{\epsilon}))^{n-1}} \right)^{1/(W+(n-1))} \\
&\leq \left(\frac{(mM)^W (2mn \cdot M)^{n-1}}{(mn \cdot M)^W (mn \cdot M)^{n-1}} \right)^{1/(W+(n-1))} \\
&\leq 2 \left(\frac{1}{n} \right)^{W/(W+(n-1))}.
\end{aligned}
$$

With increasing W, asymptotically, this ratio approaches $2/n$.

REMARK 6.1. *It is natural to ask if the asymmetric* NSW *problem is harder than the symmetric problem. As* SMatch *is the first non-trivial algorithm for the asymmetric problem, we would like to find if it gives a better approximation factor when applied to the symmetric case. However, after considerable effort, we could not resolve this question definitively. Like the above example, we could not find an example of a symmetric instance for which our analysis was tight. We conjecture that* SMatch *gives a better factor for the symmetric problem and that the symmetric case is easier than the asymmetric* NSW *case.*

7 CONCLUSIONS

In this work, we have shown two algorithms SMatch and RepReMatch. SMatch approximately maximizes the NSW for the asymmetric additive case within a factor of $O(n)$, while RepReMatch optimizes the asymmetric submodular NSW within a factor of $O(n \log n)$. We also completely resolve the submodular NSW problem for the case when there are a constant number of agents, with an $e/(e-1)$ approximation factor algorithm and a matching hardness of approximation proof.

Our algorithms also satisfy other interesting fairness guarantees for smaller special cases, namely, EF1 for the additive valuations case and a better 1.45 factor approximation for the symmetric agents with restricted additive valuations case.

Our work has initiated the investigation of the NSW problem for general cases, raising several interesting questions. Many of these questions were settled in subsequent works (Section 1.5), especially for the symmetric case. We ask if the approximation factor $O(n)$, given by SMatch, is the best possible for the asymmetric additive NSW problem. While SMatch cannot give better than a linear factor guarantee, as proved by an example in Section 6.3, there could be another algorithm with a sub-linear approximation factor.

REFERENCES

[1] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. 2017. Nash Social Welfare, Matrix Permanent, and Stable Polynomials. In *8th Innovations in Theoretical Computer Science Conference (ITCS)*. 1–12.

[2] Nima Anari, Tung Mai, Shayan Oveis Gharan, and Vijay V. Vazirani. 2018. Nash Social Welfare for Indivisible Items under Separable, Piecewise-Linear Concave Utilities. In *Proc. 29th Symp. Discrete Algorithms (SODA)*.

[3] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. 2015. Combinatorial Algorithm for Restricted Max-min Fair Allocation. In *Proc. 26th Symp. Discrete Algorithms (SODA)*. 1357–1372.

[4] Arash Asadpour and Amin Saberi. 2010. An Approximation Algorithm for Max-min Fair Allocation of Indivisible Goods. *SIAM J. Comput.* 39, 7 (2010), 2970–2989.

[5] Nikhil Bansal and Maxim Sviridenko. 2006. The Santa Claus Problem. In *Symp. Theory of Computing (STOC)*. 31–40.

[6] Siddharth Barman, Umang Bhaskar, Anand Krishna, and Ranjani G. Sundaram. 2020. Tight Approximation Algorithms for p-Mean Welfare Under Subadditive Valuations. In *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference) (LIPIcs)*, Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders (Eds.), Vol. 173. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 11:1–11:17. https://doi.org/10.4230/LIPIcs.ESA.2020.11

[7] Siddharth Barman, Anand Krishna, Pooja Kulkarni, and Shivika Narang. 2021. Sublinear approximation algorithm for nash social welfare with xos valuations. *arXiv preprint arXiv:2110.00767* (2021).

[8] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2018. Finding Fair and Efficient Allocations. In *Proc. 19th Conf. Economics and Computation (EC)*.

[9] Ivona Bezáková and Varsha Dani. 2005. Allocating indivisible goods. *ACM SIGecom Exchanges* 5, 3 (2005), 11–18.

[10] Eric Budish. 2011. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy* 119, 6 (2011), 1061–1103.

[11] Ioannis Caragiannis, David Kurokawa, Herve Moulin, Ariel Procaccia, Nisarg Shah, and Junxing Wang. 2016. The Unreasonable Fairness of Maximum Nash Welfare. In *Proc. 17th Conf. Economics and Computation (EC)*. 305–322.

[12] Suchan Chae and Herve Moulin. 2010. Bargaining among groups: an axiomatic viewpoint. *International Journal of Game Theory* 39 (2010), 71–88.

[13] Bhaskar Ray Chaudhury, Jugal Garg, and Ruta Mehta. 2021. Fair and efficient allocations under subadditive valuations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 5269–5276.

[14] Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. 2010. Dependent randomized rounding via exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 575–584.

[15] Yun Kuen Cheung, Bhaskar Chaudhuri, Jugal Garg, Naveen Garg, Martin Hoefer, and Kurt Mehlhorn. 2018. On Fair Division of Indivisible Items. In *Proc. 38th Conf. Foundations of Software Tech. and Theoret. Comp. Sci. (FSTTCS)*. 25:1–25:17.

[16] Richard Cole, Nikhil Devanur, Vasilis Gkatzelis, Kamal Jain, Tung Mai, Vijay Vazirani, and Sadra Yazdanbod. 2017. Convex Program Duality, Fisher Markets, and Nash Social Welfare. In *Proc. 18th Conf. Economics and Computation (EC)*.

[17] Richard Cole and Vasilis Gkatzelis. 2018. Approximating the Nash Social Welfare with Indivisible Items. *SIAM J. Comput.* 47, 3 (2018), 1211–1236.

[18] Vincent Conitzer, Rupert Freeman, Nisarg Shah, and Jennifer Wortman Vaughan. 2019. Group fairness for the allocation of indivisible goods. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*.

[19] Sami Davies, Thomas Rothvoss, and Yihao Zhang. 2020. A tale of Santa Claus, hypergraphs and matroids. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2748–2757.

[20] Dagmawi Mulugeta Degefu, He Weijun, Yuan Liang, Min An, and Zhang Qi. 2018. Bankruptcy to Surplus: Sharing Transboundary River Basin's Water under Scarcity. *Water Resources Management* 32 (2018), 2735–2751. Issue 8.

[21] Jugal Garg, Martin Hoefer, and Kurt Mehlhorn. 2019. Approximating the Nash Social Welfare with Budget-Additive Valuations. (2019). arxiv:1707.04428; Preliminary version appeared in the proceedings of SODA 2018.

[22] Jugal Garg, Edin Husić, Wenzheng Li, László A Végh, and Jan Vondrák. 2022. Approximating Nash Social Welfare by Matching and Local Search. *arXiv:2211.03883* (2022).

[23] Jugal Garg, Edin Husić, and László A Végh. 2021. Approximating Nash social welfare under Rado valuations. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 1412–1425.

[24] Jugal Garg, Pooja Kulkarni, and Rucha Kulkarni. 2020. Approximating Nash Social Welfare under Submodular Valuations through (Un)Matchings. In *Proc. 31st Symp. Discrete Algorithms (SODA)*. 2673–2687.

[25] Jugal Garg and Peter McGlaughlin. 2019. Improving Nash Social Welfare Approximations. In *Proc. Intl. Joint Conf. Artif. Intell. (IJCAI)*.

[26] Houba H, Van der Laan G, and Zeng Y. 2014. Asymmetric Nash solutions in the river sharing problem. *Strategic Behavior and the Environment* 4 (2014), 321–360. Issue 4.

[27] J. Harsanyi and R. Selten. 1972. A Generalized Nash Solution for Two-person Bargaining Games with Incomplete Information. *Management Science* 18 (1972), 80–106.

[28] E. Kalai. 1977. Nonsymmetric Nash Solutions and Replications of 2-person Bargaining. *International Journal of Game Theory* 6 (1977), 129–133.

[29] Frank Kelly. 1997. Charging and rate control for elastic traffic. *European Transactions on Telecommunications* 8 (1997), 33–37.

[30] Subhash Khot, Richard Lipton, Evangelos Markakis, and Aranyak Mehta. 2008. Inapproximability Results for Combinatorial Auctions with Submodular Utility Functions. *Algorithmica* 52, 1 (2008), 3–18.

[31] Subhash Khot and Ashok Kumar Ponnuswami. 2007. Approximation algorithms for the max-min allocation problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 204–217.

[32] Annick Laruellea and Federico Valenciano. 2007. Bargaining in committees as an extension of Nash's bargaining theory. *Journal of Economic Theory* 132 (2007), 291–305.

[33] Euiwoong Lee. 2017. APX-hardness of Maximizing Nash Social Welfare with Indivisible Items. *Inf. Process. Lett.* 122 (2017), 17–20.

[34] Benny Lehmann, Daniel Lehmann, and Noam Nisan. 2006. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior* 55, 2 (2006), 270–296.

[35] Wenzheng Li and Jan Vondrák. 2022. A constant-factor approximation algorithm for Nash social welfare with submodular valuations. In *2021 IEEE 62nd Annual Symposium on Foundatio ns of Computer Science (FOCS)*. IEEE, 25–36.

[36] Herve Moulin. 2003. *Fair Division and Collective Welfare*. MIT Press.

[37] John Nash. 1950. The Bargaining Problem. *Econometrica* 18, 2 (1950), 155–162.

[38] Trung Thanh Nguyen and Jörg Rothe. 2014. Minimizing envy and maximizing average Nash social welfare in the allocation of indivisible goods. *Discrete Applied Mathematics* 179 (2014), 54–68.

[39] Noam Nisan, Éva Tardos, Tim Roughgarden, and Vijay Vazirani (Eds.). 2007. *Algorithmic Game Theory*. Cambridge University Press.

[40] Zoya Svitkina and Lisa Fleischer. 2011. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM J. Comput.* 40, 6 (2011), 1715–1737.

[41] W. Thomson. 1986. Replication invariance of bargaining solutions. *Int. J. Game Theory* 15 (1986), 59–63.

[42] Hal R Varian. 1974. Equity, envy, and efficiency. *Journal of Economic Theory* 9, 1 (1974), 63 – 91.

[43] Jan Vondrák. 2008. Optimal approximation for the Submodular Welfare Problem in the value oracle model. In *Proc. 40th Symp. Theory of Computing (STOC)*. 67–74.

[44] S. Yu, E. C. van Ierland, H.-P. Weikard, and X. Zhu. 2017. Nash bargaining solutions for international climate agreements under different sets of bargaining weights. *International Environmental Agreements: Politics, Law and Economics* 17 (2017), 709–729. Issue 5.