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

Jugal Garg* jugal@illinois.edu
Pooja Kulkarni† poojark2@illinois.edu
Rucha Kulkarni‡ ruchark2@illinois.edu

Abstract

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 is known either for asymmetric agents with additive valuations or for symmetric agents beyond additive(-like) valuations.

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 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 processes that are different in both algorithms. We show that these approaches achieve approximation factors of $O(n)$ and $O(n \log n)$ for additive and submodular case respectively, which is 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.

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 following weighted geometric mean of valuations,

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

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

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 [Mou03] for additional characteristics). It was independently discovered by three different communities as a solution of the bargaining problem in classic game theory [Nas50], a well-studied notion of proportional fairness in networking [Kel97],

*University of Illinois at Urbana-Champaign. Supported by NSF CRII Award 1755619.
†University of Illinois at Urbana-Champaign. Supported by NSF CRII Award 1755619.
‡University of Illinois at Urbana-Champaign. Supported by NSF CAREER Award CCF 1750436.

[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.

and coincides with the celebrated notion of competitive equilibrium with equal incomes (CEEI) in economics [Var74]. While Nash [Nas50] only considered the symmetric case, [HS72, Kal77] proposed the asymmetric case, which has also been extensively studied, and used in many different applications, e.g., bargaining theory [LV07, CM10, Tho86], water allocation [HdLGY14, DHYZ16, DHY17, DWL+18], climate agreements [YvIWZ17], 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.[2] Efforts were then diverted to develop efficient approximation algorithms. A series of remarkable works [CG15, CDG+17, AGSS17, AMGV18, BKV18, GHM19, CCG+18] 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 [NNRR14] is known either for asymmetric agents with additive valuations or for symmetric agents beyond additive(-like) valuations. These questions are also raised in [CDG+17, BKV18].

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. [CKM+16] 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., [BKV18, CCG+18, GM19].

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 processes that are different in both algorithms. We show that these approaches achieve approximation factors of $O(n)$ and $O(n \log n)$ for additive and submodular case respectively, which is independent of the number of items. For additive valuations, our algorithm outputs an allocation that is also EF1.

1.1 Model We formally define the valuation functions we consider in this paper, and their relations to other popular 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 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, $\forall \mathcal{S} \subseteq \mathcal{G}, v_i(\mathcal{S}) = \sum_{j \in \mathcal{S}} v_i(j)$.

2. Monotone Submodular: Let $v_i(\mathcal{S}_1 \mid \mathcal{S}_2)$ denote the marginal utility of agent i for a set $\mathcal{S}_1$ of items over set $\mathcal{S}_2$, where $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{G}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. Then, the valuation function of every agent is a monotonically non decreasing function $v_i : 2^{\mathcal{G}} \to \mathbb{R}_+$ that satisfies the submodularity constraint that for all $i \in \mathcal{A}, h \in \mathcal{G}, \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{G}$,

$$v_i(h \mid \mathcal{S}_1 \cup \mathcal{S}_2) \leq v_i(h \mid \mathcal{S}_1).$$

Other popular valuation functions are budget additive (BA), separable piece-wise linear and concave (SPLC), OXS, Gross substitutes (GS), XOS and Subadditive. These function classes are related as follows.

$$\text{Additive} \subsetneq \text{SPLC} \subsetneq \text{OXS} \subsetneq \text{GS} \subsetneq \text{Submodular} \subsetneq \text{XOS},$$

$$\text{Additive} \subsetneq \text{BA} \subsetneq \text{Submodular} \subseteq \text{XOS} \subseteq \text{Subadditive}.$$

1.2 Results Table 1.2 summarizes approximation guarantees of the algorithms RepReMatch and SMatch under popular valuation functions [NTRV07], formally defined in Section 1.1. All current best known results are also stated here for reference.

To complement these results, we also provide a 1.5819-factor hardness of approximation result for the submodular NSW problem in Section 4. This hardness even applies to the case 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.

[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 [GHM19], separable piecewise linear concave (SPLC) [AMGV18], and their combination [CCG+18].

[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.

Valuations	Symmetric Agents		Asymmetric Agents	
	Hardness	Algorithm	Hardness	Algorithm
Additive	1.069 [GHM19]	1.45 [BKV18]	1.069 [GHM19]	$O(n)$ [S]
Budget-additive	—"—	1.45 [CCG+18]	—"—	—"—
SPLC	—"—	—"—	—"—	$O(n \log n)$ [R]
OXS Gross substitutes	—"—	$O(n \log n)$ [R]	—"—	—"—
Submodular	1.5819 [Thm 4.1]	—"—	1.5819 [Thm 4.1]	—"—
XOS Subadditive	—"—	$O(m)$ [NR14]	—"—	$O(m)$ [NR14]

Table 1: Summary of results. Every entry has the best known approximation guarantee for the setting followed by the reference, from this paper or otherwise, that establishes it. 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.

1.3 Techniques The main idea used in Algorithms SMatch and RepReMatch is shown in Lemma 3.1 in Section 3, which we restate in informal terms here.

Lemma (Informal). *For $k = O(n)$ and for every agent i, after removing a set S_i of k items that minimizes i's valuation for the remaining items, repeatedly matching the remaining items $\mathcal{G}\backslash(\cup_i S_i)$ to locally maximize the* NSW *objective gives every agent an allocation of value at least a $1/n$ fraction of her valuation for the remaining set of items, i.e, $v_i(\mathcal{G}\backslash(\cup_i S_i))/n$.*

When the valuation functions are additive, then such a set of k items can be efficiently found. SMatch then follows by combining these results.

It is known from [SF11] that finding a set of minimum valuation among sets of some minimum size for monotone submodular valuation functions is inapproximable within $\sqrt{m/\ln m}$ factor, where m is the number of items. Due to this, the above lemma by itself is insufficient for the submodular NSW problem. We prove Lemma 3.3 in Section 3, that implies this statement.

Lemma (Informal). *When we compute matchings between agents and items to locally maximize the weighted geometric mean of agent valuations from their matched items, then the set of items allocated in the first $\log n+1$ matchings if re-matched have a matching where every agent gets an item of value at least equal to the highest valued item from her* NSW *optimizing allocation.*

RepReMatch combines the two results in a repeated matching, un-matching and then re-matching algorithm, by applying Lemma 3.1 to the set of items that remain unallocated after a phase of $\log n+1$ matchings.

An observation used while designing both algorithms is that maximizing the logarithm of the NSW function instead of the NSW objective does not change the optimal allocation(s). Doing so allows us to maximize the (weighted) sum of the logarithms of individual agent valuations, instead of the (weighted) product of valuations. Hence, the edge weights defined in the various graphs for computing the matchings in both SMatch and RepReMatch are logarithms of agent valuations for some allocations.

Submodular NSW with constant number of agents. This is a different approach that uses techniques of maximizing submodular functions over matroids developed in [CVZ10], and a reduction of fair division problems to the problem of maximizing a submodular function over matroids from [Von08]. 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 5.2 and 5.3.

Hardness of approximation. The submodular ALLOCATION problem is to maximize the sum of valuations of agents over integral allocations of items. [KLMM08] describe 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. For

the symmetric additive NSW problem, several constant-factor approximation algorithms have been obtained. The first such algorithm used an approach based on a variant of Fisher markets [CG15], to achieve an approximation factor of 2.889. Later, the analysis of this algorithm was improved to 2 [CDG+17]. Another approach based on the theory of real stable polynomials gave an e-factor guarantee [AGSS17]. Recently, [BKV18] obtained the current best approximation factor of 1.45 using an approach based on *limited envy*. These approaches have also been extended to provide constant-factor approximation algorithms for slight generalizations of additive valuations, namely the budget-additive [GHM19], SPLC [AMGV18], and a common generalization of these two valuations [CCG+18].

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 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 [CDG+17]. However, the only known result for this general problem is an $\Omega(m)$-factor algorithm [NR14], where m is the number of items.

Two other popular welfare objectives are the social welfare where items are allocated to maximize the sum of valuations of all agents and the max-min objective where items are allocated to maximize the minimum valuation. The latter objective is also termed as the Santa Claus problem [BS06].

The social welfare problem under submodular valuations has been completely resolved with a $\frac{e}{e-1} = 1.5819$-factor algorithm [Von08] and a matching hardness result [KLMM08]. 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.

For the Santa Claus problem, extensive work has been done on the restricted additive valuations case where the value of an item j is either v_j or 0 for every agent, resulting in constant factor algorithms for the same [AKS15, DRZ18]. However, for the unrestricted additive valuations the best approximation factor is $O(\sqrt{n}\log^3 n)$ [AS10]. For the submodular Santa Claus problem, we have an $O(n)$ factor algorithm [KP07]. On the other hand, a hardness factor of 2 is the best known lower bound for both settings [BD05].

Organization of the paper: In Section 2, we describe the algorithm SMatch and analysis for the additive NSW problem. In Section 3, the corresponding discussion about the submodular NSW problem is presented. Next, Section 4 contains the hardness proof for the submodular setting. Finally, Section 5 presents the results for the special cases of submodular NSW with constant number of agents and symmetric additive NSW.

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. *Given an instance of the asymmetric additive* NSW *problem, algorithm* SMatch *returns an allocation* x *with* NSW *value at least* $1/2n$ *times the optimal objective value. That is,* $\mathsf{NSW}(\mathrm{x}) \geq \frac{1}{2n}\mathsf{OPT}$.

SMatch is a single pass algorithm that allocates up to one item to every agent per iteration such that the NSW objective is locally maximized. An issue with a naive single pass, locally optimizing greedy approach is that the initial iterations work on highly limited information. As shown in Example 2.1, such algorithms can result in outcomes with very low NSW even for symmetric agents with additive valuation functions.

Example 2.1. *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 *also values each of the remaining items at* 1*, while* B *only values the last of these at* 1*, and has* 0 *valuation for the remaining* $(m-1)$ *items. An allocation that optimizes the* NSW *of the agents will allocate the first item to* B*, and allocate all the remaining items to* A*. The optimal* NSW *objective is* $(Mm)^{1/2}$*. A repeated matching algorithm, in the first iteration, will allocate the first item to* A*, and the last to* B*. No matching can now give non zero valuation to* B*. The maximum* NSW *objective that can be generated is* $((M+\epsilon+m-1)1)^{1/2} < \sqrt{M+m}$*. Thus, using appropriate value of* M*, the ratio of* OPT *to* NSW *will depend on* m*.*

In this 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 items. 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 $\Gamma(\mathcal{A}, \mathcal{G}, \mathcal{W})$ similarly as in the submodular case, but here the edge weights are defined as $w(i,j) = \eta_i \log\left(v_i(j) + \frac{u_i}{n}\right)$, 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 $w(i,j) = \eta_i \log(v_i + v_i(j))$, 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.

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 items received by agent i in SMatch. We use $\mathbf{x}_i^* = \{g_i^1, \ldots, g_i^{\tau_i^*}\}$, τ_i and τ_i^* to denote the set of items in i's optimal bundle, and the number of items in $\mathbf{x}_i$ and $\mathbf{x}_i^*$ respectively. Then for every i, all items in $\mathbf{x}_i$ and $\mathcal{G}$ are ranked according to the decreasing utilities as per v_i. $\mathcal{G}_{i,[a:b]}$ denote 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 k^{th} 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., $u_i = \min_{\mathcal{S}\subseteq\mathcal{G}, |\mathcal{S}|\leq 2n} v_i(\mathcal{G}\setminus\mathcal{S}) = \mathcal{G}_{i,[2n+1,m]}$.[5]

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

Lemma 2.1. $v_i(h_i^t) \geq v_i(\mathcal{G}_{i,tn})$.

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 t^{th} iteration the agent will thus get an item with value at least the value of $\mathcal{G}_{i,tn}$ and the lemma follows. □

Lemma 2.2. $v_i(h_i^2, \ldots, h_i^{\tau_i}) \geq \frac{u_i}{n}$.

Proof. Using Lemma 2.1 and since $v_i(\mathcal{G}_{i,tn}) \geq v_i(\mathcal{G}_{i,tn+k})$, $\forall k \in [n-1]$

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

Thus,

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

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 $(\tau_i + 1)n \geq m$ and hence,

$$\begin{aligned} v_i(h_i^2, \ldots, h_i^{\tau_i}) &\geq \frac{1}{n}\left(v_i(\mathcal{G}_{i,[2n:m-1]})\right) \\ &\geq \frac{1}{n}\left(v_i(\mathcal{G}_{i,[2n+1:m]})\right) = \frac{1}{n} u_i. \end{aligned}$$

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

We now prove our main theorem.

Proof of Theorem 2.1.

$$\begin{aligned} \mathsf{NSW}(\mathbf{x}) &= \prod_{i=1}^{n} \left(v_i\left(h_i^1, \ldots, h_i^{\tau_i}\right)^{\eta_i}\right)^{\frac{1}{\sum_{i=1}^n \eta_i}} \\ &= \prod_{i=1}^{n} \left(\left(v_i(h_i^1) + v_i(h_i^2 \ldots, h_i^{\tau_i})\right)^{\eta_i}\right)^{\frac{1}{\sum_{i=1}^n \eta_i}} \\ &\geq \prod_{i=1}^{n} \left(\left(v_i(h_i^1) + \frac{u_i}{n}\right)^{\eta_i}\right)^{\frac{1}{\sum_{i=1}^n \eta_i}}, \end{aligned}$$

where the last inequality follows from Lemma 2.2. 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

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

Now, $u_i = \min_{\mathcal{S}\in\mathcal{G}, |\mathcal{S}|\leq 2n} v_i(\mathcal{G}\setminus\mathcal{S})$. Suppose we define, $\mathcal{S}_i^* = \arg\min_{|\mathcal{S}|\leq 2n, \mathcal{S}\subseteq\mathbf{x}_i^*} v_i(\mathbf{x}_i^* \setminus \mathcal{S})$, then $v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*) \leq u_i$. To see this, let $\mathcal{S}_i = \arg\min_{\mathcal{S}\in\mathcal{G}, |\mathcal{S}|\leq 2n} v_i(\mathcal{G}\setminus\mathcal{S})$. Now, $u_i = v_i(\mathcal{G}\setminus\mathcal{S}_i) \geq v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i) \geq v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*)$. Thus,

$$\begin{aligned} \mathsf{NSW}(\mathbf{x}) &\geq \prod_{i=1}^{n} \left(\left(\frac{1}{2n} v_i(\mathcal{S}_i^*) + \frac{1}{n} v_i(\mathbf{x}_i^* \setminus \mathcal{S}_i^*)\right)^{\eta_i}\right)^{\frac{1}{\sum_{i=1}^n \eta_i}} \\ &\geq \frac{1}{2n} \prod_{i=1}^{n} (v_i(\mathbf{x}_i^*)^{\eta_i})^{\frac{1}{\sum_{i=1}^n \eta_i}} \\ &= \frac{1}{2n} \mathsf{OPT}. \end{aligned}$$

□

[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 non-zero valuation for every item. If it does not, the other case is also straightforward and the lemma continues to hold.

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 $v_i : 2^{\mathcal{G}} \to \mathbb{R}_+$, 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.

$\mathbf{x}_i \leftarrow \emptyset, u_i \leftarrow v_i(\mathcal{G}_{i,[2n+1:m]}) \quad \forall i \in [n]$ // $\mathcal{G}_{i,[a:b]}$ defined in Section 2.2

Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}, \mathcal{W})$ with weights $\mathcal{W} = \{w(i,j) \mid w(i,j) = \eta_i \log\left(v_i(j) + \frac{u_i}{n}\right), \forall i \in \mathcal{A}, j \in \mathcal{G}\}$

Compute a maximum weight matching $\mathcal{M}$ for Γ

$\mathbf{x}_i \leftarrow \mathbf{x}_i \cup \{j \mid (i,j) \in \mathcal{M}\}, \ \forall i \in \mathcal{A}$ // allocate items according to $\mathcal{M}$

$\mathcal{G}^{rem} \leftarrow \mathcal{G} \backslash \{j \mid (i,j) \in \mathcal{M}\}$ // update set of unallocated items

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

  Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with weights $\mathcal{W} = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(j) + v_i(\mathbf{x}_i)), i \in \mathcal{A}, j \in \mathcal{G}^{rem}\}$

  Compute a maximum weight matching $\mathcal{M}$ for Γ

  $\mathbf{x}_i \leftarrow \mathbf{x}_i \cup \{j \mid (i,j) \in \mathcal{M}\}, \ \forall i \in \mathcal{A}$ // allocate items according to $\mathcal{M}$

  $\mathcal{G}^{rem} \leftarrow \mathcal{G}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\}$ // remove allocated items

end

Return $\mathbf{x}$

Remark 2.1. *When* SMatch *is applied to the instance of Example 2.1, 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 optimal* NSW.

3 Submodular Valuations

In this section we present the algorithm 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 weighted geometric mean OPT.

Theorem 3.1. *Given an instance of the asymmetric submodular* NSW *problem, algorithm* RepReMatch *returns an allocation* $\mathbf{x}$ *with* NSW *value at least* $1/(2n(\log n + 2))$ *times the optimal objective value, i.e.,*

$$\mathsf{NSW}(\mathbf{x}) \geq \frac{1}{2\mathsf{n}(\log \mathsf{n} + 2)}\mathsf{OPT}.$$

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 $\log n + 1$ iterations. We perform a similar repeated matching process in the second phase, 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 $\mathbf{x}^2_{i,t}$, in $(t+1)^{st}$ iteration, $w(i,j) = \eta_i \log(v_i(\mathbf{x}^2_{i,t} \cup \{j\}))$.

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 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. $w(i,j) = \eta_i \log(v_i(\mathbf{x}^2_i \cup \{j\}))$, where $\mathbf{x}^2_i$ 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 respectively 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 maximizes 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 $2 \le j \le \tau_i^*$, $g_i^j = \operatorname{argmax}_{g\in\mathbf{x}_i^*\setminus\{g_i^1,\ldots,g_i^{j-1}\}} v_i(g \mid \{g_i^1, \ldots, g_i^{j-1}\})$.[7]

We let $\bar{\mathbf{x}}_i^*$ be the set of items from $\mathbf{x}_i^*$ that are not allocated (to any agent) at the end of Phase 1, and 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 readability, 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 set of items allocated in Phase 2.

In the t^{th} 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. Consider the intersection of the set of items that agent i loses to other agents in the t^{th} iteration with the set of items left from her optimal bundle at the beginning of t^{th} iteration. We will refer to this set of items by $\mathcal{S}_i^t$. 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. $\mathcal{S}_i^t$ is the set of an agent's preferred items that she lost to other agents. The items that are now left are referred 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, that is, an item for which her marginal valuation (over her current allocation) is at least equal to that from her highest marginally valued attainable item.

For all i, we denote the intersection of the set of *attainable* items in the t^{th} 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 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 that the algorithm has already allocated to the agent.

Lemma 3.1. *For any $j \in [\tau_i^2 - 1]$,*

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

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, $v_i(\bar{\mathbf{x}}_{i,2}^* \mid h_i^1) \ge v_i(\bar{\mathbf{x}}_{i,2}^*) - v_i(h_i^1) = v_i(\bar{\mathbf{x}}_{i,1}^* \setminus \mathcal{S}_i^2) - v_i(h_i^1)$.

[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 : Set $\mathcal{A}$ of n agents with weights η_i, $\forall i \in \mathcal{A}$, 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:

$\mathbf{x}_i^1 \leftarrow \emptyset$, $\forall i \in \mathcal{A}$ // $\mathbf{x}_i^1$'s store the set of items allocated in Phase 1

$\mathcal{G}^{rem} \leftarrow \mathcal{G}$ // set of unallocated items before every iteration

$t \leftarrow 0$ // iteration counter

while $\mathcal{G}^{rem} \neq \emptyset$ *and* $t \leq \log n + 1$ **do**

  Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with weights $\mathcal{W} = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(j)), \forall i \in \mathcal{A}, j \in \mathcal{G}^{rem}\}$

  Compute a maximum weight matching $\mathcal{M}$ for Γ

  $\mathbf{x}_i^1 \leftarrow \mathbf{x}_i^1 \cup \{j\}, \quad \forall (i,j) \in \mathcal{M}$ // allocate items to agents according to $\mathcal{M}$

  $\mathcal{G}^{rem} \leftarrow \mathcal{G}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\}$; $t \leftarrow t+1$ // remove allocated items

end

Phase 2:

For all i, $\mathbf{x}_i^2 \leftarrow \emptyset$ // $\mathbf{x}_i^2$'s are the sets of items allocated in Phase 2

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

  Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with weights $W = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(\mathbf{x}_{i,t}^2 \cup \{j\})), \forall i \in \mathcal{A}, j \in \mathcal{G}^{rem}\}$

  Compute a maximum weight matching $\mathcal{M}$ for Γ

  $\mathbf{x}_i^2 \leftarrow \mathbf{x}_i^2 \cup \{j\}, \quad \forall (i,j) \in \mathcal{M}$ // allocate items to agents according to $\mathcal{M}$

  $\mathcal{G}^{rem} \leftarrow \mathcal{G}^{rem} \backslash \{j \mid (i,j) \in \mathcal{M}\}$ // remove allocated items

end

Phase 3:

$\mathcal{G}^{rem} \leftarrow \bigcup_i \mathbf{x}_i^1$ // release items allocated in Phase 1

Define weighted complete bipartite graph $\Gamma(\mathcal{A}, \mathcal{G}^{rem}, \mathcal{W})$ with $W = \{w(i,j) \mid w(i,j) = \eta_i \log(v_i(\mathbf{x}_i^2 \cup \{j\})), \forall i \in \mathcal{A}, j \in \mathcal{G}^{rem}\}$

Compute a maximum weight matching $\mathcal{M}$ for Γ

$\mathbf{x}_i^2 \leftarrow \mathbf{x}_i^2 \cup \{j\}, \quad \forall (i,j) \in \mathcal{M}$ // allocate items to agents according to $\mathcal{M}$

Arbitrarily allocate rest of the items to agents, let $\mathbf{x} = \{\mathbf{x}_i\}_{i \in \mathcal{A}}$ denote the final allocation

return $\mathbf{x}$

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,

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

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

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

Consider the $(r+1)^{st}$ iteration. Again, we analyze two cases.

Case 1: $h_i^r \notin \bar{\mathbf{x}}_{i,r}^*$:

$$\begin{aligned} & v_i(\bar{\mathbf{x}}_{i,r+1}^* \mid h_i^1, \ldots, h_i^r) \\ &= v_i(\bar{\mathbf{x}}_{i,r}^* \setminus \mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^r) \\ &\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^r) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^r) \\ & \qquad \text{(By submodularity of } v_i\text{)} \\ &\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^r) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}) \\ &\geq v_i(\bar{\mathbf{x}}_{i,r}^* \mid h_i^1, \ldots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \ldots, h_i^{r-1}) \\ & \quad - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \ldots, h_i^{r-1}). \\ & \qquad \text{(By monotonicity of } v_i\text{)} \end{aligned}$$

The submodularity of v_i gives the first two inequalities, and monotonicity of v_i implies the last.

Case 2: $h_i^r \in \bar{\mathbf{x}}^*_{i,r}$**:**

$$\begin{aligned}
&v_i(\bar{\mathbf{x}}^*_{i,r+1} \mid h_i^1, \dots, h_i^r)\\
&= v_i(\bar{\mathbf{x}}^*_{i,r+1} \cup \{h_i^r\} \mid h_i^1, \dots, h_i^{r-1}) -\\
&\quad v_i(h_i^r \mid h_i^1, \dots, h_i^{r-1})\\
&= v_i(\bar{\mathbf{x}}^*_{i,r} \setminus \mathcal{S}_i^{r+1} \mid h_i^1, \dots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \dots, h_i^{r-1})\\
&\geq v_i(\bar{\mathbf{x}}^*_{i,r} \mid h_i^1, \dots, h_i^{r-1}) -\\
&\quad v_i(h_i^r \mid h_i^1, \dots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \dots, h_i^{r-1}).
\end{aligned}$$

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,

$$\begin{aligned}
&v_i(\bar{\mathbf{x}}^*_{i,r+1} \mid h_i^1, \dots, h_i^r) \geq u_i^* - k_i^2 v_i(h_i^1) -\\
&\sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots, h_i^{t-1}) - v_i(h_i^1, h_i^2 \dots, h_i^{r-1}) -\\
&v_i(h_i^r \mid h_i^1, \dots, h_i^{r-1}) - v_i(\mathcal{S}_i^{r+1} \mid h_i^1, \dots, h_i^{r-1}).
\end{aligned}$$

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

$$\begin{aligned}
&v_i(\bar{\mathbf{x}}^*_{i,r+1} \mid h_i^1, \dots, h_i^r)\\
&\geq u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r-1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots, h_i^{t-1})\\
&\quad - v_i(h_i^1, h_i^2 \dots, h_i^{r-1}) - v_i(h_i^r \mid h_i^1, \dots, h_i^{r-1})\\
&\quad - k_i^{r+1} v_i(h_i^r \mid h_i^1, \dots, h_i^{r-1})\\
&= u_i^* - k_i^2 v_i(h_i^1) - \sum_{t=2}^{r} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots, h_i^{t-1})\\
&\quad - v_i(h_i^1, h_i^2 \dots, h_i^r). \qquad \square
\end{aligned}$$

The above lemma directly allows us to give a lower bound on the marginal valuation of 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 3.1. *For any* $j \in [\tau_i^2 - 1]$,

$$\begin{aligned}
&v_i(h_i^{j+1} \mid h_i^1, \dots, h_i^j)\\
&\geq \frac{1}{\bar{\tau}_i^* - \sum_{t=1}^{j+1} k_i^t} \Big(u_i^* - k_i^2 v_i(h_i^1)\\
&\qquad - \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots, h_i^{t-1})\\
&\qquad - v_i(h_i^1, h_i^2 \dots, h_i^j)\Big).
\end{aligned}$$

Proof. In any setting with a set of items $\mathcal{S} = \{s_1, \dots s_k\}$, 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 $v_i(\cdot \mid \{h_i^1, \dots, h_i^j\})$, we can say that at iteration $j+1$, h_i^{j+1} will have a marginal valuation at least,

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

Together with Lemma 3.1, 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. $\square$

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

Lemma 3.2. $v_i(h_i^1, \dots, h_i^{\tau_i^2}) \geq \frac{u_i^*}{n}$.

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 3.1, total valuation of the items obtained by agent i in Phase 2 is bounded as follows.

$$\begin{aligned}
v_i(h_i^1, \dots h_i^{\tau_i^2}) &= v_i(h_i^1, \dots, h_i^{\tau_i^2 - 1}) +\\
&\quad v_i(h_i^{\tau_i^2} \mid h_i^1, \dots, h_i^{\tau_i^2 - 1})\\
\Rightarrow v_i(h_i^1, \dots h_i^{\tau_i^2}) &\geq v_i(h_i^1, \dots, h_i^{\tau_i^2 - 1})\\
&\quad + \frac{1}{\bar{\tau_i}^* - \sum_{t=0}^{\tau_i^2 - 1} k_i^{t+1}} \Big(u_i^* - k_i^2 v_i(h_i^1)\\
&\quad - \sum_{t=2}^{\tau_i^2 - 1} k_i^{t+1} v_i(h_i^t \mid h_i^1, \dots h_i^{t-1})\\
&\quad - v_i(h_i^1, h_i^2 \dots, h_i^{\tau_i^2 - 1})\Big).
\end{aligned}$$

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 $\bar{\tau}_i^* - \sum_{t=0}^{\tau_i^2 - 1} k_i^{t+1} \leq n$, 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,

$$\begin{aligned}
v_i(h_i^1,\dots h_i^{\tau_i^2}) &\ge v_i(h_i^1,\dots,h_i^{\tau_i^2-1}) \\
&\quad + \frac{1}{n}\big(u_i^* - k_i^2 v_i(h_i^1) \\
&\qquad - \sum_{t=2}^{\tau_i^2-1} k_i^{t+1} v_i(h_i^t \mid h_i^1,\dots h_i^{t-1}) \\
&\qquad - v_i(h_i^1,h_i^2\dots,h_i^{\tau_i^2-1})\big) \\
&\ge v_i(h_i^1,\dots,h_i^{\tau_i^2-1}) \\
&\quad + \frac{1}{n}(u_i^* - (n-1)v_i(h_i^1) \\
&\qquad - \sum_{t=2}^{\tau_i^2-1} (n-1) v_i(h_i^t \mid h_i^1,\dots h_i^{t-1}) \\
&\qquad - v_i(h_i^1,h_i^2\dots,h_i^{\tau_i^2-1})) \\
&= v_i(h_i^1,\dots,h_i^{\tau_i^2-1}) \\
&\quad + \frac{1}{n}(u_i^* - (n-1)v_i(h_i^1,h_i^2\dots,h_i^{\tau_i^2-1}) \\
&\qquad - v_i(h_i^1,h_i^2\dots,h_i^{\tau_i^2-1})) \\
&= \frac{u_i^*}{n}.
\end{aligned}$$

□

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

$$\begin{aligned}
u_i^* &\le k_i^2 v_i(h_i^1) + \sum_{t=2}^{j} k_i^{t+1} v_i(h_i^t \mid h_i^1,\dots,h_i^{t-1}) \\
&\qquad + v_i(h_i^1,\dots,h_i^j) \\
&\le (n-1)v_i(h_i^1) + \sum_{t=2}^{j} (n-1) v_i(h_i^t \mid h_i^1,\dots,h_i^{t-1}) \\
&\qquad + v_i(h_i^1,\dots,h_i^j) \\
&= n \cdot v_i(h_1,\dots,h_i^j) \\
&\le n \cdot v_i(h_1,\dots,h_i^{\tau_i^2-1}),
\end{aligned}$$

which implies that Lemma 3.2 holds.

We now bound the minimum valuation that can be obtained by every agent in Phase 3. Recall that g_1^i 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 3.3. *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 released after t iterations of Phase 1, consider the set of matchings $\mathcal{M}$ where all the agents whose entire $\mathcal{G}_i^1$ is in this set of items are matched to some item from their $\mathcal{G}_i^1$s. Arbitrarily pick a matching from a subset of this set of matchings where maximum number of agents are matched to some item from their $\mathcal{G}_i^1$. Denote this matching by $\mathcal{M}^t$. Note that as for every set of agents $\mathcal{S}$ we have $|\bigcup_{i\in\mathcal{S}} \mathcal{G}_i^1| \ge |\mathcal{S}|$, in $\mathcal{M}^t$, the set of agents not matched to an item from their $\mathcal{G}_i^1$ each have at least one item from this set 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 $|\mathcal{A}_t| \le n/2^t$.

For the base case of the induction, 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 $\alpha \ge n/2$, 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 in 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 $|\mathcal{A}_1| \le n/2$.

For the inductive step, we assume the claim is true for the first t iterations. That is, for every $k \le t$, 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 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, $|\mathcal{A}_{(t+1)}| \leq n/2^{(t+1)}$. □

Proof of Theorem 3.1. From Lemma 3.2,

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

By Lemma 3.3, 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,

(3.1)
$$\mathsf{NSW}(\mathbf{x}) \geq \left(\prod_{i=1}^{n}(v_i(h_i^{1*}, h_i^2, \ldots, h_i^{\tau_i^2}))^{\eta_i}\right)^{1/(\sum_{i=1}^n \eta_i)}.$$

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 $\log n + 1$ iterations and each iteration allocates n items. Thus $|\mathbf{x}_i^* \setminus \bar{\mathbf{x}}_i^*| \leq n(\log n + 1)$ 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,

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

Thus,

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

The second inequality follows from the submodularity of valuations. The last bound, together with (3.1) gives,

$$\begin{aligned} \mathsf{NSW}(\mathbf{x}) &\geq \left(\prod_{i=1}^{n}\left(\frac{1}{2}\frac{1}{n(\log n + 2)} v_i(\mathbf{x}_i^*)\right)^{\eta_i}\right)^{1/\sum_{i=1}^n \eta_i} \\ &\geq \frac{1}{2}\frac{1}{n(\log n + 2)}\mathsf{OPT}. \end{aligned}$$

□

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 [KLMM08]. We first summarize the relevant parts of [KLMM08]. 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 colors 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 hardness of the submodular NSW problem, we note that the input of the ALLOCATION and NSW problems are the same. So let us consider the instance generated by the reduction as that of an NSW maximizing problem. From the results of [KLMM08], we can prove the following claims.

- 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 $(1-1/\mathrm{e})V$. 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 $(1-1/\mathrm{e})nV$.

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 $\frac{\mathrm{e}}{(\mathrm{e}-1)}$.

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 result are from the theory of submodular function maximization developed in [CVZ10]. 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 [CVZ10], as the main subroutine of our algorithm for the submodular NSW problem, hence first we give an overview of it, starting with a definition of the multilinear extension.

Definition 5.1 (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 5.2. *[CVZ10] 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 $V_1, \ldots, V_n \in \mathbb{R}_+$, 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, 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 5.3. *[CVZ10] 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 $(x_1, \ldots, x_n) \in P(M)$ be a point in a matroid polytope and $(X_1, \ldots, X_n) \in \{0,1\}^n$ a random solution obtained from it by randomized swap rounding. Let $\mu_0 = F(x_1, \ldots, x_n)$ and $\delta > 0$. Then*

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

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 [LLN06], and also used for approximating the submodular welfare in [Von08]. 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 functions of every agent $u_i : \{0,1\}^m \to \mathbb{R}_+$ translate naturally to submodular functions 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 gives 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 guess the optimal NSW value OPT, and the utility of every agent in the optimal allocation V_i, and check if there is an allocation X that gives every agent i a bundle of value at least (approximately) V_i. As every agent can receive at most Max utility, Max is a trivial upper bound for the maximum value of NSW, hence we perform a binary search for the optimal value in the range $(0, Max]$. Searching for sets V_i by enumerating only those sets with values that are powers of $(1+\delta)$ for some constant $\delta > 0$ will reduce the time complexity of the algorithm to $O(poly(\log(Max)/\delta))$ instead of $O(poly(Max))$, by changing the approximation factor to $(1-1/e)(1-\delta) \leq (1-1/e-\epsilon)$ for some $\epsilon > 0$.

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 5.2 ([Bud11]). *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* $v_i(\mathbf{x}_i) \geq v_i(\mathbf{x}_{\hat{i}} \backslash \{g\})$.

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 5.4. *The output of* SMatch *satisfies the* EF1 *fairness property.*

Proof. For every agent i and $j \geq 1$, the item g^i_j allocated to i in the j^{th} iteration of SMatch is valued more by i than all items $g^{i'}_k$, $k > j$ 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 j^{th} matching. Hence, $\sum_{t=1}^{j} v_i(g^i_t) \geq \sum_{t=2}^{j} v_i(g^{i'}_t)$. That is, after removing the first item $g^{i'}_1$ 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. □

Acknowledgments. We thank Chandra Chekuri and Kent Quanrud for pointing us to relevant literature in submodular function maximization theory, and having several fruitful discussions about the same.

References

[AGSS17] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash Social Welfare, Matrix Permanent, and Stable Polynomials. In *8th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 1–12, 2017.

[AKS15] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. In *Proc. 26th Symp. Discrete Algorithms (SODA)*, pages 1357–1372, 2015.

[AMGV18] Nima Anari, Tung Mai, Shayan Oveis Gharan, and Vijay V. Vazirani. Nash social welfare for indivisible items under separable, piecewise-linear concave utilities. In *Proc. 29th Symp. Discrete Algorithms (SODA)*, 2018.

[AS10] Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM J. Comput.*, 39(7):2970–2989, 2010.

[BD05] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18, 2005. Available online at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.436.18&rep=rep1&type=pdf.

[BKV18] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proc. 19th Conf. Economics and Computation (EC)*, 2018.

Algorithm 3: Approximate the Submodular NSW with constant number of 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.

For any value $Max > 0$ that is a power of $(1+\delta)$, scale all valuation functions such that $u_i(\mathcal{G}) = Max$ for all i. // Max is an upper bound on NSW objective

$OPT = Max$ // OPT is the optimal NSW objective

define $\beta > 0$, $\delta > 0$ as small positive constants

while $OPT \leq Max$ **do**

  flag=0

  for *any set in* $V = \{[V_1, V_2, \ldots, V_n] \mid \prod_i V_i = OPT, \forall i : V_i = (1+\delta)^{k_i}$ *for some* $k_i\}$ **do**

    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**

      $\mathbf{x}^* = \mathbf{x}, flag = 1$ // $flag = 1$ if current OPT value is feasible

    end

  end

  if *flag=1* **then**

    $OPT = OPT + (Max + \beta - OPT)/2$ // search if a higher value is also feasible. Adding β ensures $OPT > Max$ finally, and algorithm converges

  else

    $Max = OPT, OPT = OPT/2$ // search for a lower feasible value

  end

  $OPT =$ nearest power of $(1+\delta)$ greater than OPT

  $Max =$ nearest power of $(1+\delta)$ greater than Max

end

return $\mathbf{x}^*$

[BS06] Nikhil Bansal and Maxim Sviridenko. The Santa Claus problem. In *Symp. Theory of Computing (STOC)*, pages 31–40, 2006.

[Bud11] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

[CCG+18] Yun Kuen Cheung, Bhaskar Chaudhuri, Jugal Garg, Naveen Garg, Martin Hoefer, and Kurt Mehlhorn. On fair division of indivisible items. In *FSTTCS*, 2018.

[CDG+17] Richard Cole, Nikhil Devanur, Vasilis Gkatzelis, Kamal Jain, Tung Mai, Vijay Vazirani, and Sadra Yazdanbod. Convex program duality, Fisher markets, and Nash social welfare. In *Proc. 18th Conf. Economics and Computation (EC)*, 2017.

[CG15] Richard Cole and Vasilis Gkatzelis. Approximating the Nash social welfare with indivisible items. In *Proc. 47th Symp. Theory of Computing (STOC)*, pages 371–380, 2015.

[CKM+16] Ioannis Caragiannis, David Kurokawa, Herve Moulin, Ariel Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum Nash welfare. In *Proc. 17th Conf. Economics and Computation (EC)*, pages 305–322, 2016.

[CM10] Suchan Chae and Herve Moulin. Bargaining among groups: an axiomatic viewpoint. *International Journal of Game Theory*, 39:71–88, 2010.

[CVZ10] Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 575–584. IEEE, 2010.

[DHY17] Dagmawi Mulugeta Degefu, Weijun He, and Liang Yuan. Monotonic bargaining solution for allocating critically scarce transboundary water. *Water Resources Management*, 31:2627–2644, 2017.

[DHYZ16] Dagmawi Mulugeta Degefu, Weijun He, Liang Yuan, and Jian Hua Zhao. Water allocation in transboundary river basins under water scarcity: a cooperative bargaining approach. *Water Resources Management*, 30:4451–4466, 2016.

[DRZ18] Sami Davies, Thomas Rothvoss, and Yihao Zhang. A tale of santa claus, hypergraphs and matroids. *arXiv preprint arXiv:1807.07189*, 2018.

[DWL+18] Dagmawi Mulugeta Degefu, He Weijun, Yuan Liang, Min An, and Zhang Qi. Bankruptcy to surplus: Sharing transboundary river basin's water under scarcity. *Water Resources Management*, 32:2735–2751, 2018.

[GHM19] Jugal Garg, Martin Hoefer, and Kurt Mehlhorn. Approximating the Nash social welfare with budget-additive valuations. arxiv:1707.04428v3; Preliminary version appeared in the proceedings of SODA 2018,

2019.

[GM19] Jugal Garg and Peter McGlaughlin. Improving nash social welfare approximations. In *Proc. Intl. Joint Conf. Artif. Intell. (IJCAI)*, 2019.

[HdLGY14] Houba H, Van der Laan G, and Zeng Y. Asymmetric Nash solutions in the river sharing problem. *Strategic Behavior and the Environment*, 4:321–360, 2014.

[HS72] J. Harsanyi and R. Selten. A generalized Nash solution for two-person bargaining games with incomplete information. *Management Science*, 18:80–106, 1972.

[Kal77] E. Kalai. Nonsymmetric Nash solutions and replications of 2-person bargaining. *International Journal of Game Theory*, 6:129–133, 1977.

[Kel97] Frank Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[KLMM08] Subhash Khot, Richard Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52(1):3–18, 2008.

[KP07] Subhash Khot and Ashok Kumar Ponnuswami. Approximation algorithms for the max-min allocation problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 204–217. Springer, 2007.

[LLN06] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.

[LV07] Annick Laruellea and Federico Valenciano. Bargaining in committees as an extension of Nash's bargaining theory. *Journal of Economic Theory*, 132:291–305, 2007.

[Mou03] Herve Moulin. *Fair Division and Collective Welfare*. MIT Press, 2003.

[Nas50] John Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[NNRR14] Nhan-Tam Nguyen, Trung Thanh Nguyen, Magnus Roos, and Jorg Rothe. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Autonom. Agents & Multi-Agent Syst.*, 28(2):256–289, 2014.

[NR14] Trung Thanh Nguyen and Jörg Rothe. Minimizing envy and maximizing average Nash social welfare in the allocation of indivisible goods. *Discrete Applied Mathematics*, 179:54–68, 2014.

[NTRV07] Noam Nisan, Éva Tardos, Tim Roughgarden, and Vijay Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[SF11] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737, 2011.

[Tho86] W. Thomson. Replication invariance of bargaining solutions. *Int. J. Game Theory*, 15:59–63, 1986.

[Var74] Hal R Varian. Equity, envy, and efficiency. *Journal of Economic Theory*, 9(1):63 – 91, 1974.

[Von08] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proc. 40th Symp. Theory of Computing (STOC)*, pages 67–74, 2008.

[YvIWZ17] S. Yu, E. C. van Ierland, H.-P. Weikard, and X. Zhu. Nash bargaining solutions for international climate agreements under different sets of bargaining weights. *International Environmental Agreements: Politics, Law and Economics*, 17:709–729, 2017.