Nash Social Welfare for 2-value Instances

Hannaneh Akrami[1], Bhaskar Ray Chaudhury[2], Kurt Mehlhorn[3], Golnoosh Shahkarami[4], and Quentin Vermande[5]

[1]Max Planck Institute for Informatics, Universität des Saarlandes, hakrami@mpi-inf.mpg.de
[2]Max Planck Institute for Informatics, Universität des Saarlandes, braycha@mpi-inf.mpg.de
[3]Max Planck Institute for Informatics, mehlhorn@mpi-inf.mpg.de
[4]Max Planck Institute for Informatics, Universität des Saarlandes, gshahkar@mpi-inf.mpg.de
[5]École Normale Supérieure, Paris, quentin.vermande@ens.fr

June 29, 2021

Abstract

We study the problem of allocating a set of indivisible goods among agents with 2-value additive valuations. Our goal is to find an allocation with maximum Nash social welfare, i.e., the geometric mean of the valuations of the agents. We give a polynomial-time algorithm to find a Nash social welfare maximizing allocation when the valuation functions are *integrally 2-valued*, i.e., each agent has a value either 1 or p for each good, for some positive integer p. We then extend our algorithm to find a better approximation factor for general 2-value instances.

1 Introduction

Fair division of goods has developed into a fundamental field in economics and computer science. In a classical fair division problem, the goal is to allocate a set of goods among a set of agents in a *fair* (making every agent content with her bundle) and *efficient* (achieving good overall welfare) manner. One of the most well studied class of valuation functions are *additive* valuation functions, where the utility of a bundle is the sum of utilities of the individual goods in the bundle. When agents have additive valuation functions, the Nash social welfare or equivalently the geometric mean of the valuations, $\left(\prod_{i\in[n]} v_i(X_i)\right)^{1/n}$, is a direct indicator of the fairness and efficiency of an allocation. In particular, any allocation that maximizes Nash social welfare is *envy-free up to one good (EF1)*, i.e., no agent envies another agent following the removal of *some* single good from the other agent's bundle and *Pareto-optimal*, i.e., no allocation can give a single agent a better bundle without giving a worse bundle some other agent [10]. Unfortunately, maximizing Nash social welfare is APX-hard [21] and allocations that achieve good approximations of Nash social welfare may not have similar fairness and efficiency guarantees. Despite this, finding good approximations of Nash social welfare has received substantial interest over the years [14, 6, 3].

Although Nash social welfare maximization is hard when agents have general additive valuation functions, some special cases are polynomial time solvable. One of the interesting special cases is when agents have *binary additive valuations*, i.e., when for each agent i and each good g, we have $v_i(\{g\}) \in \{0, 1\}$. Although, this class of valuation functions seem restrictive in their expressiveness of individual preferences, several real life scenarios involve preferences that are dichotomous and as a result there is substantial research on fair division under binary valuations [1, 7, 8, 15, 16, 20]. Barman et al. [7] give a polynomial time algorithm to find an allocation with maximum Nash social welfare when agents have binary additive valuation functions. Furthermore, for binary valuations, Halpern et al. [20] show that determining a fair allocation via Nash social welfare maximization is also *strategyproof*, i.e., agents do not benefit by misreporting their preferences. A generalization of binary valuation functions are 2-value functions, where for each agent i and each good g, we have $v_i(\{g\}) \in \{a, b\}$, for some $a, b \geq 0$[1]. Amanatadis et al. [2] show that even when agents have 2-value functions, an allocation with maximum Nash social welfare implies stronger fairness notions such as envy-freeness up to any good (EFX), where no agent envies another agent following the removal of *any* single good from the other agent's bundle. Thus, even for the more general 2-value instances, finding an allocation with maximum Nash social welfare is a canonical way of dividing goods fairly and efficiently. However, Amanatadis et al. [2], leave the problem of maximizing Nash social welfare for 2-value instances open.

In this paper, we take steps towards solving this open problem. We consider the problem of maximizing Nash social welfare when the valuation functions of the agents are *integrally 2-value*, i.e., for each agent i and for each good g, we have $v_i(\{g\}) \in \{1, p\}$, for some positive integer p. The main result of our paper is a polynomial time algorithm for maximizing Nash social welfare when the valuation functions of the agents are integrally 2-value.

Theorem 10. *There exists a polynomial-time algorithm for maximizing Nash social welfare when the valuation functions of the agents are integrally 2-value.*

We remark that an immediate corollary of Theorem 10 is a better approximation of Nash social welfare when agents have 2-value functions (when p may not be integral): For any instance where p is not integral, we can round p to the closest integer ($\lceil p \rceil$ or $\lfloor p \rfloor$) and run our algorithm for integral p. It is not hard to see that this achieves a $\max\{\frac{\lfloor p \rfloor}{p}, \frac{p}{\lfloor p \rfloor}\}-$approximation of the maximum Nash social welfare. Also note that $\max\{\frac{\lfloor p \rfloor}{p}, \frac{p}{\lfloor p \rfloor}\} \geq \sqrt{2}$ which is better than the best approximation of $e^{1/e}$ known for general additive valuations [6].

We now highlight our main technical contributions.

1.1 Our Techniques

In this section, we give a brief overview of the main ideas and techniques used by our algorithm.

We define the problem of maximizing Nash social welfare as a graph problem: We have a weighted complete bipartite graph with the set of agents and the set of goods being the independent sets. The edge-weights represent the value of a good for an agent and are either 1, i.e., *light edge*, or p, i.e., *heavy edge*. We say that a good is heavy if it has at least one incident heavy edge and light otherwise. An allocation is a multi-matching in which all goods

[1] 2-value functions are binary valuation functions when $a = 0$ and $b = 1$. Additionally, for all non-binary 2-value instances, one can assume without loss of generality that $a = 1$ and $b > a$

have degree at most one and agents can have degrees larger than one. This way of defining allocation allows us to use the idea of the augmenting paths, like the algorithm in [7].

We start by mentioning some crucial structural differences of 2-value instances to binary instances. For binary instances, Halpern et al. [20] show that an allocation maximizes Nash social welfare if and only if it is *leximax*[2], i.e., the utility profile of the allocation is lexicographically maximum. However, this is not true for 2-value instances. Consider the following example: There are two agents a_1, a_2 and five goods g_1, g_2, g_3, g_4, and g_5 and $p = 5$. All goods are light for a_2 and agent a_1 values g_1 and g_2 heavily, and the other goods light. One can verify that a Nash social welfare maximizing allocation is the one where a_1 gets $\{g_1, g_2\}$ and a_2 gets $\{g_3, g_4, g_5\}$. However, this allocation is not leximax, as it is lexicographically dominated by the allocation $a_1 \leftarrow \{g_1\}$, $a_2 \leftarrow \{g_2, g_3, g_4, g_5\}$. This necessitates finding some *other tractable characterization* of a Nash social welfare maximizing allocation, in particular a characterization of the allocation of heavy goods. This motivates the main structure of our algorithm: allocate the heavy goods carefully and then allocate the light goods greedily.

Characterizing the allocation of heavy goods. The main bulk of our effort is in finding the correct allocation of the heavy goods. We briefly elaborate our technique that achieves this goal. Firstly, we give a nice characterization of the heavy goods allocation of a Nash social welfare maximizing allocation. We refer to the *heavy-part* A^H of an allocation A as the set of all heavy edges in the allocation, and we call an allocation a *heavy-only* allocation if the allocation contains only heavy edges, i.e., if $A^H = A$. One of our main structural results (shown in Theorem 1) is that there exists a Nash social welfare maximizing allocation OPT, such that the heavy-part of OPT is leximax among all heavy-only allocations of the same cardinality. Therefore, if we know the number of heavy-edges in OPT, then the utility profile of the heavy-part of OPT is unique (as it is leximax). Thus, the main question boils down to finding a heavy-only allocation, which is leximax among all heavy-only allocations of the same cardinality, and has equal number of heavy-edges as that in OPT[3].

Finding the right allocation of heavy goods. The crucial technical barrier lies in the fact that we do not know the number of heavy edges in OPT. We briefly elaborate how we overcome this connundrum. We start the algorithm by finding a heavy-only allocation A that maximizes Nash social welfare, is leximax and subject to this, has the highest number of heavy edges (such an allocation can be determined by adapting the algorithm of Barman et al. [7]). Then, we allocate the light goods greedily to A, i.e., we iterate through the unallocated light goods and allocate a light good to an agent with smallest utility. Note that by definition of A, the total number of heavy edges in A is larger than or equal to that in OPT, i.e., $|A^H| \geq |OPT^H|$. Thereafter, as our second main result, we show that the allocation A (after allocation of the light goods), exhibits *local sensitivity to the heavy edges*, i.e., if the number of heavy edges is larger than that in OPT, then a simple local reallocation can improve the Nash social welfare. In particular, given the allocation A, where A^H is leximax among all heavy-only allocations of the same cardinality, if the number of heavy edges in OPT is smaller

[2]In [20] it is called leximin. However we find leximax more expressive.

[3]At this point, we make a subtle but important clarification. Note that a Nash social welfare maximizing allocation need not allocate all heavy goods along heavy edges. Consider a simple scenario where there are two agents a_1 and a_2, and there are two goods g_1 and g_2. Both g_1 and g_2 are heavy to a_1 and both are light to a_2. Both g_1 and g_2 are heavy goods. However, in an optimal allocation, one of the heavy goods is not allocated to an agent who finds it heavy. Thus, it is not immediate how to find the number of heavy edges in OPT.

than that in A, then we can find another allocation $\hat{A}$ from A, by moving a heavy good from an agent with highest utility to an agent with lowest utility. Furthermore, we can guarantee that,

- $\hat{A}$ is leximax among all heavy-only allocations of the same cardinality,
- $|\hat{A}^H| = |A^H| - 1$, and
- the Nash social welfare of $\hat{A}$ is at least the Nash social welfare of A.

We explain how this property helps us circumvent the issue of not knowing the number of heavy edges in OPT. Our algorithm starts with allocation A. We move a heavy good from an agent with highest utility to an agent with lowest utility as long as the Nash social welfare of the allocation improves. Let our final allocation be $\tilde{A}$. Note that $|\tilde{A}^H| \leq |OPT^H|$, as otherwise we can still improve the Nash social welfare by the aforementioned reallocation. Also note that every time we perform the reallocation, the cardinality of the heavy part of the allocation decreases by exactly one. Since $|A^H| \geq |OPT^H|$ and $|\tilde{A}^H| \leq |OPT^H|$, our algorithm must have constructed an allocation $\hat{A}$ during the transition from A to $\tilde{A}$, such that $|\hat{A}^H| = |OPT^H|$. Thus, the heavy part of $\hat{A}$ is leximax among all heavy-only allocations of the same cardinality and has the number of heavy edges equal to that of OPT. Therefore, $\hat{A}$ is our desired allocation. However, since we have that Nash social welfare of $\tilde{A}$ is at least the Nash social welfare of $\hat{A}$. $\tilde{A}$ is also a Nash social welfare maximizing allocation.

1.2 Further Related Work

There are several polynomial time algorithms that find allocations achieving an $\mathcal{O}(1)$ approximation of the maximum Nash social welfare [14, 6, 3]. The algorithm by Barman et al. [6] also achieve additional properties of fairness and efficiency like approximate EF1 and approximate Pareto-optimality. The Nash social welfare maximization has $\mathcal{O}(1)$ approximation algorithms, even when agents have more general valuation functions than additive valuation functions [17, 4, 11]. When agents have submodular and subadditive valuations, algorithms with approximation factors (almost) linear in n had been obtained [18, 12, 5]. The $\mathcal{O}(n)$-approximation is also best approximation one can achieve with polynomially many value queries[4] when agents have subadditive valuations [5]. Very recently, Li and Vondrák [22] improved the approximation factor from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ when agents have submodular valuations.

There is also literature in guaranteeing high Nash social welfare with other fairness notions. For instance, relaxations of EFX can be guaranteed with high Nash welfare [9, 12], approximations of *groupwise maximin share* (GMMS) [13] and *maximin share* (MMS) [13, 10] are achieved with high Nash welfare.

1.3 Independent Work

In private communication, we are aware that similar results are obtained by Jugal Garg and Aniket Murhekar [19]. They also obtain a polynomial time algorithm to determine an allocation with maximum Nash welfare for instances where the valuation functions of the agents are integrally 2-valued.

[4] In a value query, given an agent i and a set S, the output is $v_i(S)$ where $v_i(\cdot)$ is the valuation function of agent i.

2 Preliminaries

We have a set N of n agents and a set M of m goods. Each agent i has a utility u_i. Utilities are 2-value additive, i.e., $u_i(S) = \sum_{s \in S} u_i(\{s\})$ where $u_i(\{s\}) \in \{1, p\}$ for each $i \in N$ and $s \in M$ and p is an integer greater than 1.

We maximize NSW which is the geometric mean of the utilities of agents for their bundles. Formally, for an allocation A which assigns the bundle A_i to agent i, $\text{NSW}(A) = (\Pi_{i=1}^{n} u_i(A_i))^{1/n}$. The goal is to find an allocation maximizing NSW. This notion defined by Nash [23] in 1950s captures two important properties of a desired allocation; efficiency and fairness. By $\text{NSW}(X, u)$ we mean the Nash social welfare of allocation X under utility vector u. In case u is clear from the context, we might drop it and use $\text{NSW}(X)$.

2.1 Utility Graphs

In most of the papers working on fair division, an allocation is defined as an n-tuple of disjoint bundles allocated to agents; i.e $A = (A_1, A_2, \ldots, A_n)$ where A_i is allocated to agent i. According to our techniques which heavily employ ideas similar to augmentation in matching algorithms, we find it more convenient to define an allocation from a graph point of view. Consider the complete bipartite graph $G = (N \cup M, E)$ where we have agents on one side and goods on the other side. We call the edge between agent i and good g *heavy*, if $u_i(g) = p$ and *light* otherwise. We use E^H and E^L to denote the set of *heavy* and *light* edges respectively. Moreover, good g is *heavy* for agent i if $u_i(g) = p$ and it is *light* for her otherwise. Figure 1a shows an instance with 2 agents and 3 goods.

An *allocation* is a subset A of E such that for each $g \in M$ there is at most one edge in A incident to g. Note that allocations are partial. If there is an edge $(i, g) \in A$, we say that g is assigned to i in A or i owns g in A or A assigns g to i. Otherwise, g is unassigned. An allocation is *complete* if all goods are assigned. For an agent i, we use A_i for the set of goods assigned to i in A. We refer to A_i as the bundle of i in A. Then $u_i(A_i)$ is the utility of i's bundle for i. Figure 1b shows an allocation for the instance shown in Figure 1a.

The *utility vector* of an allocation A is the vector $(u_1(A_1), \ldots, u_n(A_n))$ and its *utility profile* is the utility vector sorted in the non-descending order of utilities. A utility profile $(b_1, \ldots, b_n)$ is *lexicographically larger* than a utility profile $(c_1, \ldots, c_n)$ or $(b_1, \ldots, b_n) \succ_{lex} (c_1, \ldots, c_n)$ if the profiles are different and $b_i > c_i$ for the smallest i with $b_i \neq c_i$. An allocation A with utility profile $(a_1, \ldots, a_n)$ is *leximax* in a family $\mathcal{A}$ of allocations if for no allocation $B \in \mathcal{A}$ with utility profile $(b_1, \ldots, b_n)$, $(b_1, \ldots, b_n) \succ_{lex} (a_1, \ldots, a_n)$.

Definition 1. *(*Heavy-only allocation*) For an allocation A, its* heavy part A^H *is the restriction of A to the heavy edges, i.e., $A^H = A \cap E^H$. An allocation is* heavy-only *if $A = A^H$. Alternatively, $A \subseteq E^H$.*

For an agent i, A_i^H is the set of heavy edges incident to agent i under allocation A. We refer to $|A_i^H|$ as the *heavy degree* of i in A and denote it $\deg_H(i, A)$ or $\deg_H(i)$.

2.2 Alternating Paths

In our definition of allocations, they correspond to multi-matchings. Later, in order to improve one allocation, we alternate the edges along a path consisting of every other edge inside the current multi-matching. Hence, it is useful to define alternating paths.

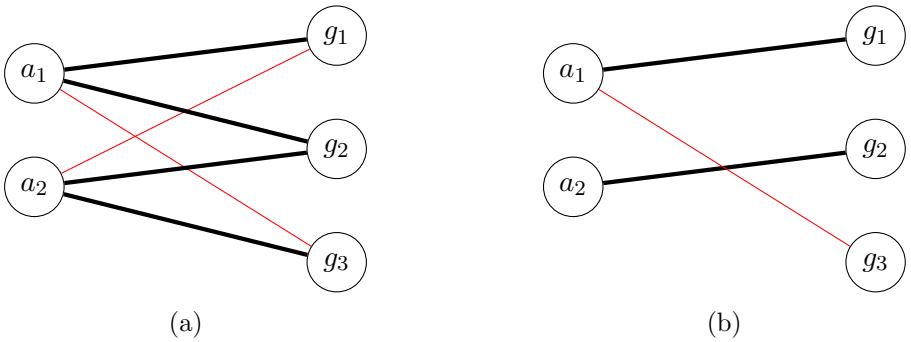

Figure 1: The graph G in (a) corresponds to an instance with 2 agents on the left side and 3 goods on the right. Thick black edges and thin red edges correspond to heavy and light edges, respectively. The graph G in (b) corresponds to a complete allocation A in which g_1 and g_3 are allocated to a_1 and g_2 is allocated to a_2. Note that this allocation does not maximize NSW. G restricted to black edges is A^H. We have $deg_H(a_1) = deg_H(a_2) = 1$.

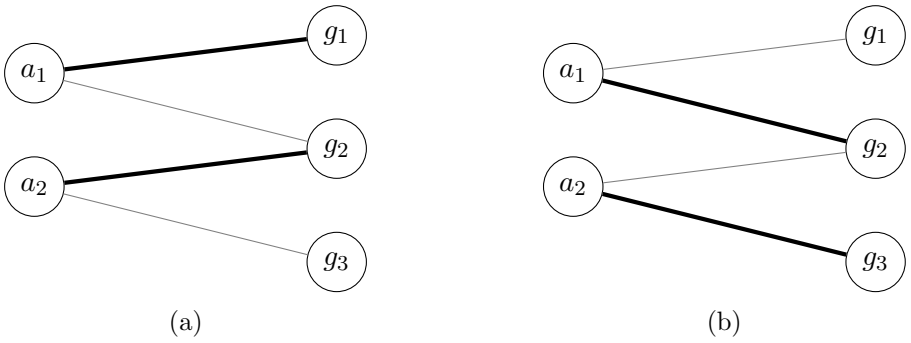

Figure 2: The path $P = (g_1, a_1, g_2, a_2, g_3)$ in (a) is a heavy alternating path where all the edges are heavy and the thick black edges shows allocation A. (b) shows $A \oplus P$.

Definition 2. *(*Heavy alternating path*) An* alternating path *with respect to an allocation A is any path whose edges are alternating between A and $E \setminus A$. A* heavy alternating path *is an alternating path all of whose edges belong to E^H.*

See Figure 2 for an example of a heavy alternating path.

Definition 3. *(*Alternating path wrt two allocations*) An* alternating path *with respect to two allocations A and B is any path whose edges are alternating between $A \setminus B$ and $B \setminus A$, i.e., between edges only in A and edges only in B.*

An *alternating path decomposition* is defined with respect to two heavy-only allocations A and B. The graph $A \oplus B$ is defined on the same set of vertices as in A and B. Moreover, the edge e appears in $A \oplus B$, if and only if e is in exactly one of A or B. We want to decompose $A \oplus B$ into edge-disjoint paths. Note that in $A \oplus B$, goods have degree zero, one, or two. For a good of degree two, the two incident edges belong to the same path. For an agent i, let a_i (b_i) be the number of A-(B)-edges incident to i in $A \oplus B$. Then we have $\min(a_i, b_i)$ alternating paths passing through i, $\max(0, a_i - b_i)$ alternating paths starting in i with an edge in A, and $\max(0, b_i - a_i)$ alternating paths starting in i with an edge in B.

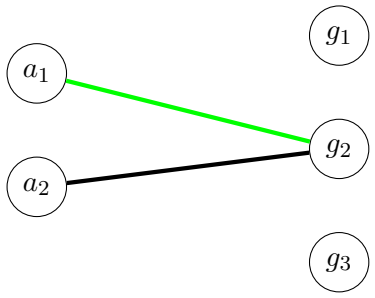

Figure 3: $A^H \oplus B^H$

If P is an even length heavy alternating path with respect to A connecting two agents i and j with the edge of P incident to i in $E^H \setminus A$ and the edge incident to j in A^H, then $A \oplus P$ contains the same number of heavy edges as A, i.e., $|A^H| = |(A \oplus P)^H| = |A^H \oplus P|$. Moreover, the heavy degree of i increased, the heavy degree of j decreased and all other heavy degrees are unchanged.

Example 1. *Consider the example shown in Figure 1a and allocation A with*

$$A^H = \{(a_1, g_1), (a_2, g_2)\}$$

shown in Figure 1b. Let B be another allocation for which

$$B^H = \{(a_1, g_1), (a_1, g_2)\}.$$

Then, Figure 3 shows $A^H \oplus B^H$. Black edges are only in A^H and green edges are only in B^H. The path decomposition of $A \oplus B$ is the only path $P = (a_1, g_2, a_1)$ that exists in this graph.

We use the following notions in the rest of the paper.

$$\begin{aligned} OPT &= \text{an allocation maximizing NSW} \\ OPT^H &= \text{the heavy part of } OPT \\ OPT_i &= \text{the bundle of agent } i \text{ in } OPT \\ OPT_i^H &= \text{the bundle of agent } i \text{ in } OPT^H \end{aligned}$$

Definition 4. *The distance of two allocations is the number of edges that only exist in one of the allocations; formally, the distance of two allocations A and B is $|A \oplus B|$.*

3 Properties of an Optimal Allocation

In this section, we study the properties of an optimal allocation. The main property is stated in Theorem 1. Roughly speaking, this theorem states that there exists an optimal allocation OPT, in which heavy goods are assigned as evenly as possible. More formally, the utility profile of OPT^H is leximax among all heavy-only allocations with the same cardinality. Later, we use this property to prove that the utility profile A^H in the end of Algorithm 2 is equal to the utility profile of OPT^H, if OPT is chosen wisely among optimal allocations. After this, it will not be difficult to prove that the utility profiles of A and OPT match.

Let $\min(OPT) = \min_i u_i(OPT_i)$ be the minimum utility of any bundle in OPT.

Lemma 1. *If $u_j(OPT_j) \geq \min(OPT) + 2$ then all goods in OPT_j are heavy for j.*

Proof. Assume otherwise, and take a good that is light for j and reallocate it to an agent i for which $u_i(OPT_i) = \min(OPT)$. This will improve NSW. □

Corollary 1. *In OPT only bundles of utility* $\min(OPT)$ *and* $\min(OPT)+1$ *can contain light goods. Bundles with higher value only contain goods that are heavy for the owner.*

Lemma 2. *There is no heavy alternating path starting with an OPT-edge from agent j to agent i if $u_j(OPT_j) > u_i(OPT_i) + p$.*

Proof. Otherwise, augmentation of the path improves NSW. □

Lemma 3. *If good g is allocated as a light good to agent i, but could be allocated as a heavy good to agent j who is allocated a light good g', then the allocation is not optimal.*

Proof. Swapping the goods g and g' among agent i and agent j, increases the value of agent j by $p-1$ and the value of agent i does not decrease. □

The rest of this section is dedicated to proving the following theorem.

Theorem 1. *Among all allocations with maximum* NSW*, there exists an allocation A such that the utility profile of A^H is leximax among all heavy-only allocations of the same cardinality.*

We choose A and heavy-only C^H as follows: (1) A is an optimal allocation, (2) C^H is leximax among all allocations of $|A^H|$ heavy goods, and (3) the distance of A^H and C^H is minimum among all allocations satisfying (1) and (2).

Let us consider $A^H \oplus C^H$. We label the edge with either A or C indicating whether it belongs to A^H or C^H. Note that in this graph, goods have degree zero, one, or two. We decompose the graph into edge-disjoint heavy alternating paths in the way that was described in section 2.2. We first show that there are no heavy alternating cycles.

Observation 2. *There are no heavy alternating cycles in the decomposition.*

Proof. Assume first that there is an alternating cycle, say D. Then $C^H \oplus D$ has the same utility profile as C^H and is closer to A^H, a contradiction. □

So we have only alternating paths. We now make more subtle observations about the edge-disjoint alternating paths in $A^H \oplus C^H$.

Alternating paths have either even or odd length. We next understand the even length alternating paths. Consider an even length alternating path, say P. The two endpoints of P have the same kind, either both are goods or both are agents. First, we show that we cannot have an even length alternating paths with both endpoints as goods.

Observation 3. *There are no even length heavy alternating paths with both endpoints as goods in the decomposition.*

Proof. If both endpoints are goods, $C^H \oplus P$ has the same utility profile as C^H and is closer to A^H, a contradiction. □

Assume next that both endpoints are agents, say i and j, and that the edge of P incident to i is in A^H and the edge of P incident to j is in C^H. Then $|A_i^H| > |C_i^H|$ and $|C_j^H| > |A_j^H|$.

First, we show that $|A_j^H| \leq |A_i^H| - 2$. Then using that, we prove that we can not have any even length alternating path with both endpoints as agents.

Observation 4. $|A_j^H| < |A_i^H|$.

Proof. By contradiction. Assume first that $|A_j^H| \geq |A_i^H|$. Then $|C_j^H| > |C_i^H| + 1$ and hence $C^H \oplus P$ is lexicographically larger than C^H, a contradiction. □

Observation 5. $|A_j^H| < |A_i^H| - 1$.

Proof. If $|A_j^H| = |A_i^H| - 1$, then $A^H \oplus P$ and A^H have the same utility profile with respect to heavy goods. Also $A^H \oplus P$ is closer to C^H than A^H. Finally, we swap the goods that are light for i in A_i with the goods that are light for j in A_j. The value of the resulting bundle for i or j is at least the value of the other agent's bundle for the other agent in A. Thus the resulting allocation is again optimal and with respect to heavy edges, it has the same utility profile as before and is closer to C^H, a contradiction. □

Observation 6. *There is no even length heavy alternating path with both endpoints as agents in the decomposition.*

Proof. By Observation 5, we have $|A_j^H| \leq |A_i^H| - 2$. Consider $A^H \oplus P$. It is closer to C^H than A^H. The value of bundle A_j went up by p and the value of bundle A_i went down by p. If A_j contains p light goods for j, move them to A_i. In this way, we obtain an allocation which has at least the NSW of A and where the allocation of heavy goods is closer to C^H. If A_j contains less than p light goods for j, then $u_i(A_i) > u_j(A_j) + p$ and A_i contains no light goods for i by Corollary 2.2. Figure 4 shows the bundles of i and j before applying $A^H \oplus P$. Let ℓ be the number of light goods allocated to j under A. We take these ℓ goods from j and allocate them to i. In the resulting allocation A' (which is shown in Figure 5), $u_i(A_i') \geq u_i(A_i) - p + \ell$ and $u_j(A_j') = u_j(A_j) + p - \ell$. Hence,

$$\begin{aligned} u_i(A_i')u_j(A_j') &\geq (u_i(A_i) - p + \ell)(u_j(A_j) + p - \ell) \\ &= u_i(A_i)u_j(A_j) + (p - \ell)(u_i(A_i) - u_j(A_j) - (p - \ell)) \\ &> u_i(A_i)u_j(A_j). \end{aligned}$$

The last inequality holds since $p > \ell$ and $u_i(A_i) > u_j(A_j) + p$. Since the bundles of other agents are not changed, $\mathrm{NSW}(A') > \mathrm{NSW}(A)$ which is a contradiction. □

We now come to the case that we have no even length alternating path. Next we want to show that there is no odd length alternating path.

Observation 7. *There is no odd length heavy alternating path in the decomposition.*

Proof. Since $|A^H| = |C^H|$, if there is an odd length alternating path, there must be two odd length alternating path P and Q, where P starts and ends with an edge in A^H and Q starts and ends with an edge in C^H. The paths P and Q are edge-disjoint. For each path one of the endpoints is an agent and one is a good. Let i be the agent endpoint of P and j be the agent endpoint of P. Then $|A_i^H| > |C_i^H|$ and $|C_j^H| > |A_j^H|$.

We now argue as above but with the even length alternating path replaced by $P \cup Q$. □

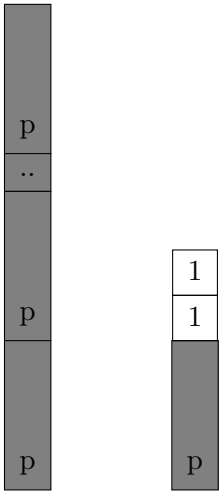

Figure 4: Bundles of agents i and j before applying $A^H \oplus P$.

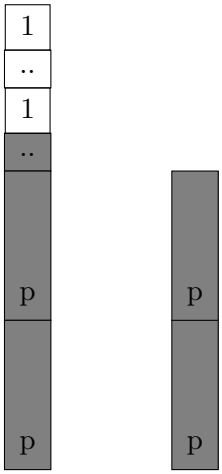

Figure 5: Bundles of agents i and j after applying $A^H \oplus P$ and moving the mentioned light goods. Note that in case any of these goods is not light for i, $u_i(A_i)$ will be even larger, resulting in an allocation with even better NSW.

Proof of Theorem 1. Consider$A^H \oplus C^H$. By Observation 2, we do not have any heavy alternating cycle. Also, by Observations 3 and 6, there is no even length heavy alternating path. Moreover, by Observation 7, there is no odd length heavy alternating path. Hence, we have $A^H = C^H$, and therefore, A^H is leximax among all heavy-only allocation of the same cardinality. □

Corollary 2. *Among all allocations maximizing* NSW*, let OPT be such that OPT^H is leximax among the heavy parts of allocations maximizing* NSW*. OPT^H is leximax among all heavy-only allocations B^H with $|B^H| = |OPT^H|$.*

Lemma 4. *Let OPT be an optimal allocation. Then the following allocation is also optimal. Start with OPT^H and then allocate the goods in OPT^L greedily, i.e., allocate the goods one by one and for each good $g \in OPT^L$ choose an arbitrary agent i for which $u_i(OPT'_i) = \min(OPT')$, add edge (i, g) to the current allocation OPT', and update OPT'.*

Proof. Consider a sequence of assigning the goods in OPT^L which results in OPT and is closest to greedily assigning a good to an agent with minimum utility. Assume that at some point with partial allocation X, we assign a good g to an agent i but there is another agent j with minimum utility such that $u_j(X_j) < u_i(X_i)$. Note that $u_i(X_i \cup \{g\}) > u_j(X_j) + 1$ and i has a light good. By Corollary 1, $u_i(OPT_i) - u_j(OPT_j) \leq 1$. Hence, after assigning g to i, a light good h should be assigned to j as well. First assigning h to j and then g to i makes the sequence closer to a greedy sequence which is a contradiction. Therefore, $u_i(X_i)$ is minimum and the sequence is in fact greedily assigning goods to agents with minimum utility. □

Before we go on to explain the algorithm, it is worth mentioning why our approach does not work when p is not an integer. Theorem 1 is not true if p is half-integer, say $p = 3/2$. Consider an instance with two agents, two goods that are heavy for both agents and three goods that are light for both agents. In the optimal allocation, both agents have bundles of value 3. The bundle of one agent contains the two goods that are heavy for her and the bundle of the other agents contains three goods that are all light for her. The heavy part of this allocation is not leximax among all allocations in which two goods are allocated as heavy goods.

4 Algorithm

In this section we elaborate the algorithm. Our algorithm operates in 3 phases. The first phase finds a heavy-only allocation which maximizes the NSW. This phase is equivalent to maximizing NSW in a binary instance. Barman et al. [7] proved that this is possible in polynomial time.

In the second phase, we greedily allocate the remaining goods (one by one) to an agent with minimum utility. Note that all these goods are light for all agents. Otherwise, the output of the first phase does not maximize NSW among all heavy-only allocations. This is why this phase is called "allocating light goods".

The third phase is the most technical one in which we reshuffle some of the goods. More precisely, we take a heavy good from the bundle of an agent with maximum utility and allocate it to an agent with minimum utility as long as NSW increases. We later show in Lemma 6 that the reallocated goods are light for their new owners. This means, as long as there is a progress, we turn some of the heavy goods into light goods.

Phase 1: Heavy-Only Allocations

As a first step we concentrate on heavy-only allocations. We first show how to compute a heavy-only allocation maximizing NSW. We use our own words to describe Algorithm 1 in [7].

In order to compute a heavy-only allocation maximizing NSW, we start with a heavy-only allocation A of maximum cardinality, i.e., in A any good that is heavy for at least one agent is assigned to an agent for which it is heavy. We then improve the NSW of A by augmentation of some heavy alternating paths. As long as there is a heavy even-length alternating path P connecting agent i to agent j, starting with an edge outside A and ending with an edge in A, and with the heavy degree of j at least two larger than the heavy degree of i in A, we augment P to A, i.e., we update A to $A \oplus P$. When the process stops, A maximizes NSW.

Algorithm 1 BinaryMaxNSW

Input : $N, M, v = (u_1, ..., u_n)$
Output: allocation A

```
let G be the corresponding graph
find the maximum multi-matching A
while there is an alternating path a_0, g_1, ..., g_k, a_k s.t |A_0| ≤ |A_k| − 2 do
    for ℓ ← k to 1 do
        A ← A\(a_ℓ, g_ℓ)
        A ← A ∪ (a_{ℓ−1}, g_ℓ)
return A
```

Barman et al. showed in [7] that Algorithm 1 outputs an allocation with maximum NSW. Furthermore, Halpern et al. proved in [20] that in binary instances the set of leximax allocations is identical to the set of allocations with maximum NSW.

Theorem 8. *Having an instance with only heavy edges as an input, Algorithm 1 outputs an allocation with maximum Nash Social Welfare. Furthermore, an optimal allocation OPT is leximax and hence the utility profile of the optimal allocation is unique.*

Before proceeding to the next phase, we briefly explain how to get leximax heavy-only (partial) allocations of different cardinalities. The heavy-only allocation maximizing NSW is a maximum cardinality heavy-only allocation. In order to compute heavy-only allocations of smaller cardinality, we repeatedly remove an edge from A. We take any bundle A_i with $u_i(A_i) = \max_j u_j(A_j)$ and remove an edge of A incident to i. In this way, we will obtain optimal allocations for every cardinality.

Lemma 5. *Let $C^H \subseteq E^H$ be leximax among all allocations $D^H \subseteq E^H$ with $|D^H| = |C^H|$. Let $(c_1, c_2, \ldots, c_n)$ be the utility profile of C^H and let t be such that $c_{t-1} < c_t = \cdots = c_n$. Then allocation $\hat{C}^H$ with utility profile $(\hat{c}_1, \ldots, \hat{c}_n) = (c_1, \ldots, c_{t-1}, c_t - p, c_{t+1}, \ldots, c_n)$, is leximax among all heavy-only allocations with the same cardinality.*

Proof. Let $\hat{D}^H$ be leximax among all allocations with $|\hat{C}^H|$ many heavy allocated goods. Let $(\hat{d}_1, \ldots, \hat{d}_n)$ be the utility profile of $\hat{D}^H$. Note that since $|\hat{D}^H| = |\hat{C}^H| < |C^H|$, there exists a good g that is unallocated under $\hat{D}^H$ and is of value p for some agent a.

Consider the smallest i such that $\hat{d}_i \neq \hat{c}_i$. Then $\hat{d}_i > \hat{c}_i$ since $\hat{D}^H$ is leximax. If $i < t$, allocating g to agent a results in an allocation D^H with $|D^H| = |C^H|$ which is lexicographically larger than C^H. This contradicts the choice of C^H.

So $\hat{d}_i = \hat{c}_i$ for all $i < t$ and hence $\Sigma_{i=t}^{n} \hat{d}_i = \Sigma_{i=t}^{n} \hat{c}_i = (n-t+1)c_n - 1$. Since $(c_n - 1, c_n, \ldots, c_n)$ is leximax among all $(n-t+1)$ tuples with sum $(n-t+1)c_n - 1$, $(\hat{d}_t, \ldots, \hat{d}_n) \preceq_{lex} (c_n - 1, \ldots, c_n) = (\hat{c}_t, \ldots, \hat{c}_n)$. Hence $\hat{C}^H$ is leximax among all allocations $\hat{D}^H$ with $|\hat{D}^H| = |\hat{C}^H|$. □

Corollary 3. *Let $(p \cdot a_1, ..., p \cdot a_n)$ and $(p \cdot b_1, ..., p \cdot b_n)$ be the utility profile of heavy-only allocations A and B. Note that $\Sigma_{i=1}^{n} a_i = |A^H|$ and $\Sigma_{i=1}^{n} b_i = |B^H|$ and let $|A^H| \leq |B^H|$. If the utility profile of A is leximax among all the utility profiles of heavy-only allocations C with $|C| = |A|$ and same holds for B, then for all $1 \leq i \leq n : a_i \leq b_i$.*

Proof. Keep removing goods from the bundle with maximum utility and minimum index in B until we reach an allocation $\hat{B}$ with $|\hat{B}^H| = |A^H|$. By Lemma 5, $(p \cdot \hat{b}_1, ..., p \cdot \hat{b}_n) \succeq_{lex} (p \cdot a_1, ..., p \cdot a_n)$ and therefore $(\hat{b}_1, ..., \hat{b}_n) = (a_1, ..., a_n)$. The fact that $\hat{b}_i \leq b_i$ for all $i \in [n]$, completes the proof. □

Phase 2: Allocating Light Goods

As long as there is an unallocated good, allocate it to an agent with minimum utility.

Phase 3: Increasing NSW

As long as NSW increases, take a good from an agent with maximum utility and give it to an agent with minimum utility. See Algorithm 2.

Algorithm 2 TwoValueMaxNSW

Input : $N, M, u = (u_1, ..., u_n)$
Output: allocation A

\\ Phase 1
let $u'_i : 2^M \to \mathbb{N}$ be an additive function and $u'_i(g) = \lfloor u_i(g)/p \rfloor$ for all agents i and all goods g
$u' \leftarrow (u'_1, \ldots, u'_n)$
$A =$BinaryMaxNSW(N, M, u')
\\ Phase 2
while there is an unallocated good g **do**
    let $u_1(A_1) \leq u_2(A_2) \leq ... \leq u_n(A_n)$
    let k be the maximum index s.t $u_k(A_k) = u_1(A_1)$
    $A \leftarrow A \cup \{(k, g)\}$
\\ Phase 3
while $u_n(A_n) > p \cdot u_1(A_1) + p$ **do**
    let k be the maximum index s.t $u_k(A_k) = u_1(A_1)$
    let t be the minimum index s.t $u_t(A_t) = u_n(A_n)$
    let g be a good such that $(t, g) \in A$
    $A \leftarrow A \backslash \{(t, g)\}$
    $A \leftarrow A \cup \{(k, g)\}$
return A

5 Correctness

Phase 1 already gives us an optimal allocation X of the heavy only goods which is also leximax on the allocation of the heavy only goods. In phase 2, we allocate the small valued goods as "evenly" as possible. The only reason why our solution may not be optimal is that the number of heavy goods in an optimal allocation Y may be less than that in X. However, if this is the case, then we can move from X to Y by making small local improvements in Nash social welfare by moving heavy goods from one bundle to the other (captured by Theorem 9 and Corollary 4).

Let A be the allocation that Algorithm 2 outputs. First we prove that there is an allocation OPT with maximum NSW such that the utility profile of OPT^H and A^H are the same. Then we prove that in allocation OPT, the remaining goods are allocated the same way as in A.

We start with showing some invariants of Algorithm 2.

Lemma 6. *Fix a numbering of the agents at the beginning of phase 3 such that $u_1(A_1) \leq u_2(A_2) \leq \ldots \leq u_{n-1}(A_{n-1}) \leq u_n(A_n)$. During phase 3, the following holds.*

a. The ordering $u_1(A_1) \leq u_2(A_2) \leq \ldots \leq u_{n-1}(A_{n-1}) \leq u_n(A_n)$ is maintained.

b. If A_i contains a good that is light for i, then $u_i(A_i) \leq u_1(A_1) + 1$.

c. A^H is leximax among all heavy-only allocations of the same cardinality.

d. Whenever a good is moved in phase 3, say from bundle A_t to bundle A_k, all goods in A_t are heavy for t and light for k.

Proof. We prove statements a) to d) by induction on the number of iterations in phase 3. Before the first iteration a) and d) trivially hold. Claim b) holds since in phase 2 we allocate only goods that are light for every agent and since the next good is always added to a lightest bundle. Claim c) holds by Theorem 8.

Assume now that a) to c) hold before the i-th iteration and that we move a good g from A_t to A_k in iteration i. We will show that d) holds for A_t and A_k and that a) to c) hold after iteration i.

By the condition of the while-loop, we have $u_t(A_t) > p \cdot (u_k(A_k) + 1)$. Thus A_t contains only goods that are heavy for t by part b) of the induction hypothesis. Let g be any good in A_t. If we also have $u_k(g) = p$, then moving g from A_t to A_k would result in an allocation of heavy goods that is lexicographically larger, a contradiction to c). Thus $u_k(g) = 1$.

After moving g, c) holds by lemma 5. Note that g is given from an agent with maximum utility and is not heavy for its new owner.

Since k is the largest index such that $u_k(A_k) = u_1(A_1)$ before the i-th iteration, b) holds after the i-th iteration.

It remains to show that part a) holds after the i-th iteration. The weight of the k-th bundle increases by 1 and the weight of the t-bundle decreases by p. We need to show $u_t(A_t) \geq u_{t-1}(A_{t-1}) + p + \delta$, where $\delta = 1$ if $k = t - 1$ and $\delta = 0$ otherwise.

- If $k = t - 1$, we have $u_t(A_t) \geq p \cdot (u_{t-1}(A_{t-1}) + 1) + 1$ and hence $u_t(A_t) - u_{t-1}(A_{t-1}) - p - 1 \geq (p - 1) \cdot u_{t-1}(A_{t-1}) \geq 0$.

- If $k < t-1$, by definition of t, $u_t(A_t) > u_{t-1}(A_{t-1})$. If all goods in A_{t-1} are heavy for $t-1$, the difference in weight is at least p and we are done. If A_{t-1} contains a good that is light for $t-1$, then $u_{t-1}(A_{t-1}) \leq u_k(A_k) + 1$ by condition b) and hence $u_t(A_t) \geq p \cdot u_{t-1}(A_{t-1}) + 1$. This implies $u_t(A_t) \geq u_{t-1}(A_{t-1}) + p$ except if $u_{t-1}(A_{t-1}) = 0$. In the latter case, $k = t-1$, a case we have already dealt with.

We also need to show that after moving the good, $u_k(A_k) \leq u_{k+1}(A_{k+1})$. By the choice of k and the fact that $u_i(X_i) \in \mathbb{N}$, $u_k(A_k) \leq u_{k+1}(A_{k+1}) + 1$ holds before moving the good. After moving the good, by condition d), $u_k(A_k)$ increases by 1 and therefore, $u_k(A_k) \leq u_{k+1}(A_{k+1})$. □

Theorem 9. *Let OPT be an allocation that maximizes* NSW *and subject to that, maximizes $|OPT^H|$. Let A be the output of algorithm 2. Then $|OPT^H| \geq |A^H|$.*

Proof. Assume $|OPT^H| < |A^H|$. By the choice of OPT, A cannot maximize NSW. We first show that we may assume $|OPT_i^H| \leq |A_i^H|$ for all i. We can obtain a leximax heavy-only allocation C^H of cardinality $|OPT^H|$ from A^H by repeatedly removing a good from the lowest indexed bundle of maximum utility. The utility profiles of C^H and OPT^H agree and hence there is a bijection π of the set of agents such that $|C_i^H| = |OPT_{\pi(i)}^H|$. Let ℓ_i be the number of goods in $OPT_{\pi(i)}^H$ which are light for $\pi(i)$. Note that the number of goods which are not allocated under C^H is equal to the number of light goods under OPT, i.e, $\Sigma_{i\in[n]}\ell_i$. Obtain an allocation C from C^H by giving ℓ_i not yet allocated goods to C_i. Then $u_i(C_i) \geq u_{\pi(i)}(OPT_{\pi(i)})$ for all i. Thus, C is optimal and $u_i(C_i) = u_{\pi(i)}(OPT_{\pi(i)})$. Also $|C_i| \leq |A_i^H|$ for all i. We may therefore assume $|OPT_i^H| \leq |A_i^H|$ for all i.

Since C is optimal, Lemma 4 gives us an alternative way of obtaining an optimal allocation. Start from C^H and then allocate the goods that are allocated as light goods (i.e. the goods that are light for their owner) in OPT in a greedy fashion. Let then R be the set of agents from which we removed a good in moving from A^H to C^H. Note that no good is added to the bundle of an agent in R when adding goods greedily. Otherwise, we would have a contradiction to Lemma 3.

Since A is not optimal there must be an agent k such that $u_k(OPT_k) > u_k(A_k)$. The bundle OPT_k must contain a good g that is light for k.

Since A is the output of Algorithm 2, moving a good from A_n to A_1 does not increase NSW. So

$$(u_1(A_1) + 1)(u_n(A_n - p)) \leq u_1(A_1)u_n(A_n)$$

and hence,

$$u_n(A_n) \leq p \cdot u_1(A_1) + p.$$

Consider $OPT^H \oplus A^H$ and its alternating path decomposition. Since the number of heavy edges in OPT is less than the number of heavy edges in A, there must be alternating path P starting with an edge in A and ending in an edge in A. Let t and g be the endpoints of the path; t is an agent and g is a good. Then g must be allocated in OPT as a light good to an agent j since g has degree one in $OPT^H \oplus A^H$.

Since $t \in R$, $|OPT_t^H| < |A_t^H|$, and no good is added to OPT_t in the greedy assignment of goods, $u_t(OPT_t) + p \leq u_t(A_t) \leq u_n(A_n)$. We also have $u_k(OPT_k) > u_k(A_k) \geq u_1(A_1)$. Therefore, we get

$$u_t(OPT_t) + p \leq u_n(A_n) \leq p \cdot u_1(A_1) + p \leq p \cdot u_k(OPT_k)$$

and hence,

$$u_t(OPT_t) \leq p \cdot u_k(OPT_k) - p.$$

Taking g from j's bundle and changing OPT to $OPT \oplus P$ increases the number of heavy goods allocated to t by one. In case $k \neq j$, we replenish j from OPT_k by taking a light good from OPT_k and allocating it to OPT_j. This reallocation of goods does not decrease NSW as:

$$(u_k(OPT_k) - 1)(u_t(OPT_t) + p) - u_k(OPT_k)u_t(OPT_t) = p \cdot u_k(OPT_k) - u_t(OPT_t) - p \geq 0. \tag{1}$$

For the new allocation $\widehat{OPT}$ we have $\text{NSW}(\widehat{OPT}) \geq \text{NSW}(OPT)$ and $|\widehat{OPT}^H| > |OPT^H|$ which contradicts the choice of OPT. □

Lemma 7. *Let $\hat{A}$ be the partial allocation after phase 1 of the Algorithm 2. Then $|\hat{A}^H| \geq |OPT^H|$ for any optimal allocation OPT.*

Proof. Assume otherwise. Then there should be a heavy edge (i, g) which is not in $\hat{A}$. After allocating g to agent i, NSW increases. This contradicts Theorem 8. □

By Lemma 7 and Theorem 9, we can assume in some round in phase 3 of Algorithm 2 with allocation $\tilde{A}$, $|\tilde{A}^H| = |OPT^H|$. Then, by Lemma 6.c and Theorem 1, we get the following Corollary.

Corollary 4. *There is an optimal allocation OPT such that $\tilde{A}^H = OPT^H$.*

So far, we have proved that considering only heavy allocated goods in $\tilde{A}$ and OPT, we end up having the same utility profile. By Lemma 6.a and Lemma 4, in both allocations OPT and $\tilde{A}$, light goods are allocated as evenly as possible.

So we can conclude that the utility profiles of $\tilde{A}$ and OPT are equal. In each round of the phase 3 of Algorithm 2, Nash Social Welfare increases. This means $\text{NSW}(A) \geq \text{NSW}(\tilde{A}) = \text{NSW}(OPT)$.

Theorem 10. *There exists a polynomial-time algorithm for the Nash Social Welfare problem with 2-value instances 1 and p when p is an integer.*

Proof. We already proved that the output of Algorithm 2 is a Nash social welfare maximizing allocation. It only remains to prove that this algorithm is polynomial-time. By [7], Algorithm 1 and hence the first phase of Algorithm 2 runs in polynomial time. The second phase clearly takes polynomial time. By lemma 6.d, the number of heavy goods under A is decreasing after each iteration of the third phase. Therefore, this phase can be run at most m times. All in all, Algorithm 2 terminates in polynomial time. □

6 Approximation for General Two Value Instances

In this section, we want to introduce an approximation algorithm for general 2-value instances. Therefore we can assume that the values are 1 and any real value $p > 1$. The idea is to round p to $\lfloor p \rfloor$ or $\lceil p \rceil$ to have an integer value and then run Algorithm 2. We will show that this results in an approximation factor of $\max\{\frac{\lfloor p \rfloor}{p}, \frac{p}{\lceil p \rceil}\}$ for NSW. Since $\max\{\frac{\lfloor p \rfloor}{p}, \frac{p}{\lceil p \rceil}\} \geq \sqrt{2}$, we improve the state-of-the-art $e^{1/e}$ approximation of the maximum Nash social welfare in 2-value instances.

Algorithm 3 ApproximationTwoValueMaxNSW

Input : $N, M, u = (u_1, ..., u_n)$
Output: allocation $APXAlg$

1: **if** $\frac{\lfloor p \rfloor}{p} \geq \frac{p}{\lceil p \rceil}$ **then**:
2: let $u'_i : 2^M \to \mathbb{N}$ be an additive function and $u'_i(g) = \lfloor u_i(g) \rfloor$ for all agents i
3: **else**:
4: let $u'_i : 2^M \to \mathbb{N}$ be an additive function and $u'_i(g) = \lceil u_i(g) \rceil$ for all agents i
5: $u' \leftarrow (u'_1, \ldots, u'_n)$
6: $APXAlg =$TwoValueMaxNSW(N, M, u')
7: return $APXAlg$

Theorem 11. *There exists a polynomial-time* $\max\{\frac{\lfloor p \rfloor}{p}, \frac{p}{\lceil p \rceil}\}-$*approximation algorithm for the Nash Social Welfare problem with 2-value instances* 1 *and* $p > 1$.

Proof. First assume $\frac{\lfloor p \rfloor}{p} \geq \frac{p}{\lceil p \rceil}$. Consider OPT which is the optimum allocation under utility vector u. Consider this allocation under utility vector u'. The utility of the goods are either not changed, or are multiplied by $\lfloor p \rfloor / p$. Therefore,

$$\text{NSW}(OPT, u') \geq \text{NSW}(OPT, u) \cdot \frac{\lfloor p \rfloor}{p}.$$

Since $p \geq \lfloor p \rfloor$ and Algorithm 2 gives the optimum allocation under utility vector u', we have

$$\text{NSW}(APXAlg, u) \geq \text{NSW}(APXAlg, u') \geq \text{NSW}(OPT, u').$$

Therefore we can conclude,

$$\text{NSW}(APXAlg, u) \geq \text{NSW}(OPT, u) \cdot \frac{\lfloor p \rfloor}{p}.$$

Now consider the second case in which $\frac{\lfloor p \rfloor}{p} \leq \frac{p}{\lceil p \rceil}$. Consider allocation $APXAlg$ under utility vector u. With comparison to the utility vector u', the utility of the goods are either not changed, or are multiplied by $\lceil p \rceil / p$. Therefore,

$$\text{NSW}(APXAlg, u) \geq \text{NSW}(APXAlg, u') \cdot \frac{p}{\lceil p \rceil}.$$

Now consider OPT which is the optimum allocation under utility vector u. Since Algorithm 2 gives the optimum allocation under u' and $\lceil p \rceil \geq p$, we have

$$\text{NSW}(APXAlg, u') \geq \text{NSW}(OPT, u') \geq \text{NSW}(OPT, u).$$

Therefore we can conclude,

$$\mathrm{NSW}(APXAlg, u) \geq \mathrm{NSW}(OPT, u) \cdot \frac{p}{\lceil p \rceil}.$$

□

References

[1] M. Aleksandrov, H. Aziz, S. Gaspers, and T. Walsh. Online fair division: analysing a food bank problem. *CoRR*, abs/1502.07571, 2015.

[2] G. Amanatidis, G. Birmpas, A. Filos-Ratsikas, A. Hollender, and A. A. Voudouris. Maximum nash welfare and other stories about EFX. *Theor. Comput. Sci.*, 863:69–85, 2021.

[3] N. Anari, S. O. Gharan, A. Saberi, and M. Singh. Nash Social Welfare, Matrix Permanent, and Stable Polynomials. In *8th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 1–12, 2017.

[4] N. Anari, T. Mai, S. O. Gharan, and V. V. Vazirani. Nash social welfare for indivisible items under separable, piecewise-linear concave utilities. In *Proc. 29th Symp. Discrete Algorithms (SODA)*, pages 2274–2290, 2018.

[5] S. Barman, U. Bhaskar, A. Krishna, and R. G. Sundaram. Tight approximation algorithms for p-mean welfare under subadditive valuations. In *ESA*, volume 173 of *LIPIcs*, pages 11:1–11:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[6] S. Barman, S. K. Krishnamurthy, and R. Vaish. Finding fair and efficient allocations. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, pages 557–574, 2018.

[7] S. Barman, S. K. Krishnamurthy, and R. Vaish. Greedy algorithms for maximizing Nash social welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 7–13, 2018.

[8] S. Bouveret and M. Lemaître. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. In *Autonomous Agents and Multi-Agent Systems (AAMAS) 30, 2*, pages 259–290, 2016.

[9] I. Caragiannis, N. Gravin, and X. Huang. Envy-freeness up to any item with high Nash welfare: The virtue of donating items. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*, pages 527–545, 2019.

[10] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang. The unreasonable fairness of maximum Nash welfare. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 305–322, 2016.

[11] B. R. Chaudhury, Y. K. Cheung, J. Garg, N. Garg, M. Hoefer, and K. Mehlhorn. On fair division for indivisible items. In *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 25:1–25:17, 2018.

[12] B. R. Chaudhury, J. Garg, and R. Mehta. Fair and efficient allocations under subadditive valuations. In *AAAI*, pages 5269–5276. AAAI Press, 2021.

[13] B. R. Chaudhury, T. Kavitha, K. Mehlhorn, and A. Sgouritsa. A little charity guarantees almost envy-freeness. In *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*, pages 2658–2672, 2020.

[14] R. Cole and V. Gkatzelis. Approximating the nash social welfare with indivisible items. *SIAM J. Comput.*, 47(3):1211–1236, 2018.

[15] A. Darmann and J. Schauer. Maximizing Nash product social welfare in allocating indivisible goods. *Europ. J. Oper. Res.*, 247(2):548–559, 2015.

[16] R. Freeman, S. Sikdar, R. Vaish, and L. Xia. Equitable allocations of indivisible goods. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 280–286. ijcai.org, 2019.

[17] J. Garg, M. Hoefer, and K. Mehlhorn. Approximating the Nash social welfare with budget-additive valuations. In *Proc. 29th Symp. Discrete Algorithms (SODA)*, 2018.

[18] J. Garg, P. Kulkarni, and R. Kulkarni. Approximating Nash social welfare under submodular valuations through (un)matchings. In *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*, pages 2673–2687, 2020.

[19] J. Garg and A. Murhekar. Private communication. 2021.

[20] D. Halpern, A. D. Procaccia, A. Psomas, and N. Shah. Fair division with binary valuations: One rule to rule them all. In *WINE*, volume 12495 of *Lecture Notes in Computer Science*, pages 370–383. Springer, 2020.

[21] E. Lee. APX-hardness of maximizing Nash social welfare with indivisible items. *Inf. Process. Lett.*, 122:17–20, 2017.

[22] W. Li and J. Vondrák. A constant-factor approximation algorithm for nash social welfare with submodular valuations. *CoRR*, abs/2103.10536, 2021.

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