# Approximation Algorithms for the Weighted Nash Social Welfare via Convex and Non-Convex Programs

Adam Brown*[1], Aditi Laddha †[2], Madhusudhan Reddy Pittu ‡[3], and Mohit Singh*[1]

[1]Georgia Institute of Technology
[2]Yale University
[2]Carnegie Mellon University

**Abstract**

In an instance of the weighted Nash Social Welfare problem, we are given a set of $m$ indivisible items, $\mathcal{G}$, and $n$ agents, $\mathcal{A}$, where each agent $i \in \mathcal{A}$ has a valuation $v_{ij} \geq 0$ for each item $j \in \mathcal{G}$. In addition, every agent $i$ has a non-negative weight $w_i$ such that the weights collectively sum up to 1. The goal is to find an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ that maximizes $\prod_{i \in \mathcal{A}} \left( \sum_{j \in \sigma^{-1}(i)} v_{ij} \right)^{w_i}$. When all the weights equal to $\frac{1}{n}$, the problem reduces to the classical Nash Social Welfare problem, which has recently received much attention. In this work, we present a $5 \cdot \exp\left(2 \cdot D_{\mathrm{KL}}(\mathbf{w} \,||\, \frac{\vec{1}}{n})\right) = 5 \cdot \exp\left(2 \log n + 2 \sum_{i=1}^{n} w_i \log w_i\right)$-approximation algorithm for the weighted Nash Social Welfare problem, where $D_{\mathrm{KL}}(\mathbf{w} \,||\, \frac{\vec{1}}{n})$ denotes the KL-divergence between the distribution $w$ and the uniform distribution on $[n]$.

We generalize the convex programming relaxations for the symmetric variant of Nash Social Welfare presented in [CDG+17, AGSS17] to two different mathematical programs. The first program is convex and is necessary for computational efficiency, while the second program is a non-convex relaxation that can be rounded efficiently. The approximation factor derives from the difference in the objective values of the convex and non-convex relaxation.

## 1 Introduction

In an instance of the weighted Nash Social Welfare problem, we are given a set of $m$ indivisible items $\mathcal{G}$, and a set of $n$ agents, $\mathcal{A}$. Every agent $i \in \mathcal{A}$ has a weight $w_i \geq 0$ such that $\sum_{i \in \mathcal{A}} w_i = 1$ and an additive valuation function $\mathbf{v}_i : 2^{\mathcal{G}} \to \mathbb{R}_{\geq 0}$. Let $v_{ij} := \mathbf{v}_i(\{j\})$. The goal is to find an assignment of items, $\sigma : \mathcal{G} \to \mathcal{A}$ so that the following welfare function is maximized:

$$\prod_{i \in \mathcal{A}} \left( \sum_{j \in \sigma^{-1}(i)} v_{ij} \right)^{w_i}. \tag{1.1}$$

For ease of notation, we will work with the log objective and denote

$$\mathrm{NSW}(\sigma) = \sum_{i \in \mathcal{A}} w_i \log \left( \sum_{j \in \sigma^{-1}(i)} v_{ij} \right) \tag{1.2}$$

and $\mathrm{OPT} = \max_{\sigma : \mathcal{G} \to \mathcal{A}} \mathrm{NSW}(\sigma)$ denote the optimal log objective. The case in which $\mathbf{w} = \mathbf{u}$ where $u_i = \frac{1}{n}$ for each $i \in \mathcal{A}$ is the much-studied symmetric Nash social welfare problem, where the objective is the geometric mean of agents' valuations.

*ajmbrown@gatech.edu, msingh94@gatech.edu; supported in part by NSF CCF-2106444 and NSF CCF-1910423.
†aditi.laddha@yale.edu; supported in part by NSF CCF-2007443.
‡mpittu@andrew.cmu.edu; supported in part by NSF awards CCF-1955785 and CCF-2006953

Fair and efficient division of resources among agents is a fundamental problem arising in various fields [BT05, BT96, BCE+16, RW98, Rot15, You94]. While there are many social welfare functions which can be used to evaluate the efficacy of an assignment of goods to the agents, the Nash Social Welfare function is well-known to interpolate between fairness and overall utility. The symmetric Nash Social Welfare function first appeared as the solution of an arbitration scheme proposed by Nash for two-person bargaining games that was generalized to multiple players [NJ50, KN79]. Since then, it has been widely used in numerous fields to model resource allocation problems. An attractive feature of the objective is that it is invariant under scaling by any of the agent's valuations and therefore each of the agents can specify their utility in their own units (see [CM04] for a detailed treatment). While the theory of Nash Social Welfare objective was initially developed for divisible items, more recently it has been applied in the context of indivisible items. We refer the reader to [CKM+19] for a comprehensive overview of the problem in the latter setting. Indeed, optimizing the Nash Social Welfare objective also implies notions of fairness such as *envy free* allocation in an approximate sense [CKM+19, BKV18].

The Nash Social Welfare function with weights (also referred to as asymmetric or non symmetric Nash Social Welfare) was first studied in the seventies [HS72, Kal77] in the context of two person bargaining games. For example, in the bargaining context, it allows different agents to have different weights. This flexibility has made the problem arise in many different domains, including bargaining theory [CM04, LV07], water resource allocation [FKL12, HLZ13], and climate agreements [YIWZ17]. From a context of indivisible goods, the study of this problem has been much more recent [GKK20, GHV21, GHL+23]. In this work, we aim to shed light on this problem, especially, with a focus on mathematical programming relaxations for the problem.

**1.1 Our Results and Contributions** In this work, we present a $\exp\left(2\log 2 + \frac{1}{2e} + 2D_{\mathrm{KL}}(\boldsymbol{w}\,||\,\boldsymbol{u})\right) \approx 4.81 \cdot \exp\left(2\log n - 2\sum_{i=1}^{n} w_i \log \frac{1}{w_i}\right)$-approximation algorithm for the weighted Nash Social Welfare problem with additive valuations. When all the weights are the same, this gives a constant factor approximation. Our algorithm builds on and extends a convex programming relaxation for the symmetric variant of Nash Social Welfare presented in [CG15, CDG+17, AGSS17]. In the theorem, we state the guarantee in log objective and therefore, the guarantee becomes an additive one.

THEOREM 1.1. *Let $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$ be an instance of the weighted Nash Social Welfare problem with $\sum_{i\in\mathcal{A}} w_i = 1$ and $|\mathcal{A}| = n$ agents. There exists a polynomial time algorithm (Algorithm 1) that given $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$ returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

$$\mathrm{NSW}(\sigma) \geq \mathrm{OPT} - 2\log 2 - \frac{1}{2e} - 2 \cdot D_{\mathrm{KL}}(\mathbf{w}\,||\,\mathbf{u}),$$

*where* OPT *is the optimal log-objective and $D_{KL}(\mathbf{w}||\mathbf{u}) = \log n - \sum_{i\in\mathcal{A}} w_i \log \frac{1}{w_i}$.*

We additionally note that our approach can be modified to give $\exp\left(2\log 2 + \frac{1}{2e} + D_{\mathrm{KL}}(\boldsymbol{w}\,||\,\boldsymbol{u})\right)$-approximation that runs in pseudo-polynomial time. The details of this result can be found in the full version of the paper. Observe that the KL-divergence term $D_{\mathrm{KL}}(\mathbf{w}\,||\,\mathbf{u}) = \left(\log n - \sum_{i\in\mathcal{A}} w_i \log \frac{1}{w_i}\right)$ is always upper bounded by $\log(nw_{\max})$ which is exactly the guarantee of previous work [GHL+23]. In many settings, the term $D_{\mathrm{KL}}(\mathbf{w}\,||\,\mathbf{u})$ can be significantly smaller than $nw_{\max}$; for example, consider the setting where $w_1 = \frac{1}{\log n}$ and $w_i = \frac{1}{n-1}(1 - \frac{1}{\log n})$ for $i = 2, \ldots, n$, i.e., one agent has significantly higher weight than other. In this case, our results imply $O(1)$-approximation while previous results imply $O(\frac{n}{\log n})$-approximation. A calculation of this can be found in Example A in Appendix A.

Our algorithm relies on two mathematical programming relaxations for the weighted Nash Social Welfare problem both of which generalize the convex relaxation for the symmetric version [CDG+17, CG15, AGSS17]. One of the relaxations is non-convex but retains a lot of structural insights obtained for the convex relaxation in the symmetric version. We show that the same rounding algorithm as in the symmetric version [CG15] gives a $O(1)$-approximation for the weighted version when applied to a fractional solution of the non-convex program. While the rounding algorithm is the same, the analysis requires new ideas as many of the interpretations via market equilibrium in the symmetric case are no longer present in the weighted version. Unfortunately, due to its non-convex nature, we cannot solve this relaxation to optimality even though it can be rounded efficiently. Now the second mathematical programming relaxation comes to the rescue. This relaxation is convex and thus can be

solved efficiently. We solve the convex relaxation, then use the non-convex relaxation to measure the change in objective as we process the solution and eventually round to an integral assignment. The approximation factor of $D_{\mathrm{KL}}(\mathbf{w} \,||\, \mathbf{u})$ arrives due to the difference in objective values of these two programs. In Section 1.3, we give a technical overview before giving the details in the later sections.

### 1.2 Preliminaries

**KL-Divergence.** For two probability distributions $\mathbf{p}, \mathbf{q}$ over the same discrete domain $\mathcal{X}$, the KL-divergence between $\mathbf{p}$ and $\mathbf{q}$ is defined as

$$D_{\mathrm{KL}}(\mathbf{p} \,||\, \mathbf{q}) = \sum_{x \in \mathcal{X}} p(x) \log \left( \frac{p(x)}{q(x)} \right) .$$

It is well-known, via Gibb's inequality, that the KL-divergence between two distributions is non-negative and is zero if and only if $\mathbf{p}$ and $\mathbf{q}$ are identical.

We use this fact crucially in the following claim.

CLAIM 1.1. *Given positive $z_1, \ldots, z_d$, for any $y_1, y_2, \ldots, y_d \geq 0$,*

$$\sum_{j=1}^{d} y_j \log \left( \sum_{j=1}^{d} z_j \right) - \sum_{j=1}^{d} y_j \log \left( \sum_{j=1}^{d} y_j \right) \geq \sum_{j=1}^{d} y_j \log z_j - \sum_{j=1}^{d} y_j \log y_j .$$

*Proof.* Define vectors $\mathbf{y} = (y_1, \ldots, y_d)$ and $\mathbf{z} = (z_1, \ldots, z_d)$. Let $\widetilde{\mathbf{y}} = \frac{\mathbf{y}}{\|\mathbf{y}\|_1}$ and $\widetilde{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|_1}$. The inequality reads $D_{\mathrm{KL}}(\widetilde{\mathbf{y}} \,||\, \widetilde{\mathbf{z}}) \geq 0$. ∎

Moreover, if $q$ is a uniform distribution on $\mathcal{X}$ and $p$ an arbitrary distribution on the same domain, then

$$D_{\mathrm{KL}}(p \,||\, q) = \log |\mathcal{X}| - \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}.$$

**Matching Polytope.** Consider a complete bipartite graph $G = (\mathcal{G} \cup \mathcal{A}, E)$ where $E$ contains an edge $(i, j)$ for each $i \in \mathcal{A}$ and $j \in \mathcal{G}$. Let $\mathcal{M}(\mathcal{A})$ denote the set of all matchings in $G$ of size $|\mathcal{A}|$, i.e., matchings which have an edge incident to every vertex in $\mathcal{A}$. Then the convex hull $\mathcal{M}(\mathcal{A})$ is given by

$$\begin{aligned} & \mathbf{b} \in \mathbb{R}_{\geq 0}^{|\mathcal{A}| \times |\mathcal{G}|} \\ & \sum_{j \in \mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\ & \sum_{i \in \mathcal{A}} b_{ij} \leq 1 \quad \forall j \in \mathcal{G} . \end{aligned} \tag{1.3}$$

We call this polytope the Agent Matching Polytope of $(\mathcal{A}, \mathcal{G})$.

### 1.3 Technical Overview

Building on the algorithm of [CG15], [CDG+17] introduced the following relaxation for the symmetric Nash Social Welfare problem.

$$\begin{aligned} \max_{b,q} \quad & \frac{1}{n} \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} b_{ij} \log (v_{ij}) - \frac{1}{n} \sum_{j \in \mathcal{G}} \left( \sum_{i \in \mathcal{A}} b_{ij} \right) \log \left( \sum_{i \in \mathcal{A}} b_{ij} \right) \\ \text{s.t.} \quad & \sum_{j} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\ & \sum_{i} b_{ij} \leq 1 \quad \forall j \in \mathcal{G} \\ & b_{ij} \geq 0 \quad \forall (i, j) \in \mathcal{A} \times \mathcal{G} \end{aligned} \tag{CVX-Sym}$$

They showed that (CVX-Sym) is a convex relaxation of the Nash Social Welfare objective, and the prices used by the algorithm presented in [CG15] can be obtained as dual variables of (CVX-Sym). The convex relaxation is,

interestingly, not in the assignment variables. Indeed given an optimal assignment $\sigma : \mathcal{G} \to \mathcal{A}$, the corresponding setting of the variables $b_{ij}$ is

$$b_{ij} = \begin{cases} \frac{v_{ij}}{\sum_{k \in \sigma^{-1}(i)} v_{ik}} & \text{if } \sigma(j) = i \\ 0 & \text{otherwise}. \end{cases} \tag{1.4}$$

One can verify that $\mathbf{b}$ satisfies all the constraints in (CVX-Sym) and its objective equal to the log of the geometric means of the valuations.

**Programs for Weighted NSW.** While [CDG+17] and [CG15] used intuition from economics and market equilibrium to arrive at (CVX-Sym), these concepts do not generalize to the case when weights are not uniform. We take a different, more algebraic approach that relates the use of log-concave polynomials in [AGSS17] to the convex programming interpretation given in [CDG+17]. More concrete details on this relationship and how it leads to the two programs can be found in the full version of the paper.

By setting $\mathbf{b}$ to be the same value as (1.4), it is natural to see that (CVX-Weighted) is a relaxation of weighted NSW. However, this program is not easy to round. To circumvent this issue, we introduce an intermediate non-convex mathematical program in (NCVX-Weighted), which is also a relaxation of weighted NSW.

$$\begin{aligned} \max_{\mathbf{b}} \quad & f_{\text{cvx}}(\mathbf{b}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} \\ & - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left( \sum_{i \in \mathcal{A}} w_i \, b_{ij} \right) \\ & + \sum_{i \in \mathcal{A}} w_i \log w_i \\ \text{s.t.} \quad & \sum_{j \in \mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\ & \sum_{i \in \mathcal{A}} b_{ij} \leq 1 \quad \forall j \in \mathcal{G} \\ & b_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G} \end{aligned}$$

(CVX-Weighted)

$$\begin{aligned} \max_{\mathbf{b}} \quad & f_{\text{ncvx}}(\mathbf{b}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} \\ & - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left( \sum_{i \in \mathcal{A}} b_{ij} \right) \\ \text{s.t.} \quad & \sum_{j \in \mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\ & \sum_{i \in \mathcal{A}} b_{ij} \leq 1 \quad \forall j \in \mathcal{G} \\ & b_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G} \end{aligned}$$

(NCVX-Weighted)

LEMMA 1.1. *(CVX-Weighted) and (NCVX-Weighted) are relaxations of the weighted Nash Social Welfare problem. Moreover, When the weights are symmetric, i.e., $w_i = 1/n$ for all $i \in \mathcal{A}$, the programs (CVX-Weighted) and (NCVX-Weighted) are equivalent to the convex program (CVX-Sym).*

We formally prove Lemma 1.1 in Appendix A.

Note that the constraints for both (CVX-Weighted) and (NCVX-Weighted) are identical. We use $\mathcal{P}(\mathcal{A}, \mathcal{G})$ to denote this constraint polytope.

DEFINITION 1.1. (FEASIBILITY POLYTOPE) *For a set of $m$ indivisible items $\mathcal{G}$, and a set of $n$ agents, $\mathcal{A}$, the feasibility polytope, denoted by $\mathcal{P}(\mathcal{A}, \mathcal{G})$ is defined as*

$$\mathcal{P}(\mathcal{A}, \mathcal{G}) := \left\{ \mathbf{b} \in \mathbb{R}_{\geq 0}^{|\mathcal{A}| \times |\mathcal{G}|} \; : \; \sum_{j \in \mathcal{G}} b_{ij} = 1 \; \forall i \in \mathcal{A} \, , \sum_{i \in \mathcal{A}} b_{ij} \leq 1 \; \forall j \in \mathcal{G} \right\}.$$

*The constraint $\sum_{j \in \mathcal{G}} b_{ij} = 1$ is called the Agent constraint for agent $i$ and the contraint $\sum_{i \in \mathcal{A}} b_{ij} \leq 1$ is referred to as the Item constraint for item $j$.*

We will refer to points in $\mathcal{P}(\mathcal{A},\mathcal{G})$ as either feasible points or solutions.

Observe that $\mathcal{P}(\mathcal{A},\mathcal{G})$ is identical to the Agent Matching polytope as defined in equation (1.3). So $\mathcal{P}(\mathcal{A},\mathcal{G})$ is the convex hull of the set of matchings between $\mathcal{A}$ and $\mathcal{G}$ in which every agent is matched.

**The two relaxations.** We observe that the objective of (CVX-Weighted) is a concave function, and thus it is a polynomial time tractable convex program. But the objective of (NCVX-Weighted) is not necessarily concave. Despite this, (NCVX-Weighted) still satisfies many desirable properties: given a point $\mathbf{b} \in \mathcal{P}(\mathcal{A},\mathcal{G})$, one can efficiently find another point $\widetilde{\mathbf{b}} \in \mathcal{P}(\mathcal{A},\mathcal{G})$ without decreasing the objective $f_{\text{ncvx}}$ such that the graph formed by support of $\widetilde{\mathbf{b}}$ is a forest, as stated in the following lemma. We formally define the support graphs in Definition 2.1.

LEMMA 1.2. *Let $\overline{\mathbf{b}}$ be any feasible point in $\mathcal{P}(\mathcal{A},\mathcal{G})$. Then there exists an acyclic solution, $\mathbf{b}^{\text{forest}}$, in the support of $\overline{\mathbf{b}}$ such that*

$$f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{ncvx}}(\overline{\mathbf{b}})\,.$$

*Such a $\mathbf{b}^{\text{forest}}$ can be found in time polynomial in $|\mathcal{A}|$ and $|\mathcal{G}|$.*

Next, we establish that one can efficiently round any feasible point whose support graph is a forest to an assignment.

THEOREM 1.2. *For a Nash Social Welfare instance $(\mathcal{A},\mathcal{G},\mathbf{v},\mathbf{w})$, given a vector $\mathbf{b} \in \mathcal{P}(\mathcal{A},\mathcal{G})$ such that the support of $\mathbf{b}$ is a forest, there exists a deterministic polynomial time algorithm (Algorithm 2) which returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

$$\text{NSW}(\sigma) \geq f_{\text{cvx}}(\mathbf{b}) - D_{\text{KL}}(\boldsymbol{w} \,||\, \boldsymbol{u}) - 2\log 2 - \frac{1}{2e}\,.$$

Our rounding algorithm is the same as that of [CG15], however our analysis is quite different. Our analysis relies crucially on two facts: the relative stability of stationary points of (CVX-Weighted) and the interplay between the values of $f_{\text{cvx}}$ and $f_{\text{ncvx}}$. First, we establish that any stationary point of (CVX-Weighted) is relatively stable, i.e., the difference between the objective values of a stationary point and any feasible solution is independent of the valuations $\mathbf{v}$, and therefore can be bounded effectively. Second, we show that for any feasible solution, the difference between $f_{\text{cvx}}$ and $f_{\text{ncvx}}$ is at most the KL-Divergence between the weights and the constant vector.

We use the stability of stationary points of (CVX-Weighted) along with the structure of the feasiblility polytope to iteratively sparsify a stationary solution to obtain a matching between the agents and bundles of items while only losing a constant factor in the objective. It is worth noting that the first term in the objective $f_{\text{ncvx}}$ (and $f_{\text{cvx}}$) is linear in the variable $\mathbf{b}$. As the constraint set on $\mathbf{b}$ is a matching polytope, the solution optimizing a linear objective would be a matching in which all agents receive exactly one item. While such a matching would be very suboptimal compared to OPT, our algorithm constructs an augmented graph which indeed contains a matching with value comparable to OPT. The crux of our algorithm is to find a feasible vector in the matching polytope for which $f_{\text{ncvx}}$ is close to OPT and the additional non-linear term in $f_{\text{ncvx}}$ are relatively small.

The remaining challenge to our approach is that (NCVX-Weighted) is not a convex program and therefore we cannot efficiently find a global optima that maximizes $f_{\text{ncvx}}(\mathbf{b})$. However, we show that the objective of (NCVX-Weighted) and (CVX-Weighted) differ by at most the $D_{\text{KL}}(\mathbf{w} \,||\, \mathbf{u})$, as stated in the following lemma. We leverage this fact to initialize (NCVX-Weighted) with the globally optimal solution of (CVX-Weighted) to obtain the approximation guarantee.

LEMMA 1.3. *For any $\mathbf{b} \in \mathcal{P}(\mathcal{A},\mathcal{G})$ and weights $w_1,\ldots,w_n > 0$ with $\sum_{i\in\mathcal{A}} w_i = 1$,*

$$0 \leq f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) \leq D_{\text{KL}}(\boldsymbol{w} \,||\, \boldsymbol{u}) = \log n - \sum_{i\in\mathcal{A}} w_i \log \frac{1}{w_i}\,.$$

By combining Lemma 1.2, Theorem 1.2, and Lemma 1.3 we obtain our main result.

**1.4 Related Work** The problem of finding the allocation which maximizes the Nash Social Welfare objective is an NP-hard problem, as was proven by [NNRR14]. Additionally, [Lee17] showed that finding such an allocation

is also APX-hard. From an algorithmic perspective, the first constant-factor approximation for the symmetric version was provided in [CG15] using analogies from market equilibrium. [AMGV18] provided an improved analysis of the algorithm from [CG15, CDG+17] and introduced a convex programming relaxation. Using an entirely different approach, [AGSS17] also provided a constant factor approximation for the symmetric variant, where their analysis employed the theory of log-concave polynomials. The best-known approximation factor with linear valuations of 1.45 is due to [BKV18], where they provide a pseudopolynomial time algorithm that finds an allocation which is envy-free up to one good. Their algorithm is entirely combinatorial and is polynomial time when the valuations are bounded.

Another setting of interest is when the valuation of each agent is submodular instead of additive. For instance, [GHV21] gave a constant factor approximation algorithm for maximizing the symmetric Nash Social welfare function when the agents' valuations are Rado, a special subclass of submodular functions. In the weighted case, the approximation factor of this algorithm depends on the ratio of the maximum weight to the minimum weight. [LV22] provided a constant factor approximation algorithm for the symmetric case with submodular valuations. More recently, [GHL+23] gave a local search-based algorithm to obtain an $O(nw_{\max})$-approximation for the weighted case and a 4-approximation for the symmetric case with submodular valuations. Note that this $O(nw_{\max})$-approximation factor was also the previously best-known approximation for the weighted case, even when considering additive valuations.

## 2 Approximation Algorithm

Before describing our algorithm, we need the following definitions.

Definition 2.1. (Support Graph) *For a vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, the support graph of $\mathbf{b}$, denoted by $G_{\text{supp}}(\mathbf{b})$, is a bipartite graph with vertex set $\mathcal{A} \cup \mathcal{G}$. For any $i \in \mathcal{A}$ and $j \in \mathcal{G}$, the edge $(i, j)$ belongs to the edge set of $G$ if and only if $b_{ij} > 0$.*

Definition 2.2. (Acyclic Solution) *A vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ is called an acyclic solution if the support graph of $\mathbf{b}$, $G_{\text{supp}}(\mathbf{b})$, does not contain any cycles.*

For ease of notation, given any feasible point $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, we use variables $\mathbf{q} \in \mathbb{R}^{|\mathcal{G}|}$ to denote the projection of $\mathbf{b}$ to $\mathcal{G}$, i.e.,

$$q_j := \sum_{i \in \mathcal{A}} b_{ij}$$

for each $j \in \mathcal{G}$. Since $\mathbf{q}$ is completely defined by $\mathbf{b}$, with abuse of notation, we will interchangeably use $\mathcal{P}(\mathcal{A}, \mathcal{G})$ to denote feasible vectors $\mathbf{b}$ as well as $(\mathbf{b}, \mathbf{q})$. Similarly, we will use $f_{\text{ncvx}}(\mathbf{b}, \mathbf{q})$ and $f_{\text{cvx}}(\mathbf{b}, \mathbf{q})$, to also denote the objective $f_{\text{ncvx}}(\mathbf{b})$ and $f_{\text{cvx}}(\mathbf{b})$, respectively. With a slight abuse of notation, we define

$$f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log q_j \, .$$

for any $\mathbf{b} \in \mathbb{R}^{|A| \times |G|}$ and $\mathbf{q} \in \mathbb{R}^{|G|}$.

Our main algorithm, Algorithm 1, begins by finding the optimal solution $\overline{\mathbf{b}}$ to the convex program (CVX-Weighted). It then constructs another feasible point, $\mathbf{b}^{\text{forest}}$, in support of $\overline{\mathbf{b}}$ such that the support graph of $\mathbf{b}^{\text{forest}}$ is a forest. While this may decrease the value $f_{\text{cvx}}$, we show that it is always possible to find $\mathbf{b}^{\text{forest}}$ such that $\mathbf{b}^{\text{forest}}$ has the same $f_{\text{ncvx}}$ objective value as $\overline{\mathbf{b}}$. Finally, the algorithm then rounds this solution, $\mathbf{b}^{\text{forest}}$, to an integral solution using Algorithm 2. Theorem 1.2 establishes a bound on the rounding error incurred during Algorithm 2.

**Algorithm 1:** Approximation Algorithm for Weighted Nash Social Welfare

**Input**. NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$
$\overline{\mathbf{b}} \leftarrow$ optimal solution of (CVX-Weighted)
$\overline{\mathbf{q}} \leftarrow$ vector in $\mathbb{R}^{|\mathcal{G}|}$ with $\overline{q}_j = \sum_{i \in \mathcal{A}} \overline{b}_{ij}$
$(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}}) \leftarrow$ acyclic solution in support of $\overline{\mathbf{b}}$ such that $f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{ncvx}}(\overline{\mathbf{b}})$
$\sigma \leftarrow$ output of Algorithm 2 with input $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w}, \mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$
**Output.** $\sigma$

Lemma 1.2, which we re-state below for the reader's convenience, guarantees the existence of an acyclic solution in the support of which does not decrease the $f_{\text{ncvx}}$ function, ensuring that the algorithm is well-defined. It is worth mentioning that for the unweighted case, the existence of an acyclic optimum was utilized by [CG15, CDG+17] for the convex program (CVX-Sym). In the weighted setting, this structural property is not inherited by the convex program (CVX-Weighted) but is inherited by the non-convex program (NCVX-Weighted).

LEMMA 1.2. *Let $\overline{\mathbf{b}}$ be any feasible point in $\mathcal{P}(\mathcal{A},\mathcal{G})$. Then there exists an acyclic solution, $\mathbf{b}^{\text{forest}}$, in the support of $\overline{\mathbf{b}}$ such that*

$$f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{ncvx}}(\overline{\mathbf{b}})\,.$$

*Such a $\mathbf{b}^{\text{forest}}$ can be found in time polynomial in $|\mathcal{A}|$ and $|\mathcal{G}|$.*

*Proof.* Let $G_{\text{supp}}(\overline{\mathbf{b}})$ contain a cycle $(i_0, j_0, i_1, \ldots, j_{\ell-1}, i_\ell)$ with $i_0 = i_\ell$, where $i_x \in \mathcal{A}$ and $j_y \in \mathcal{G}$. The main idea is to modify the variables $\bar{\mathbf{b}}$ on this cycle while ensuring the value of $\bar{\mathbf{q}}$ does not change. If $\bar{\mathbf{q}}$ is fixed, then $f_{\text{ncvx}}(\cdot, \bar{\mathbf{q}})$ is linear in the input, and as a result, we can *cancel* the cycle by considering the following vector. Define $\boldsymbol{\delta} \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{G}|}$ with $\delta_{i_x j_x} := 1$ and $\delta_{i_{x+1} j_x} := -1$ for $x \in \{0, \ldots, \ell-1\}$, and $\delta_{ij} := 0$ otherwise.

Note that $\sum_{i\in\mathcal{A}} \delta_{ij} = 0$ for any item $j$. As a result, for each $j \in \mathcal{G}$,

$$\sum_{i\in\mathcal{A}} \bar{b}_{ij} + \varepsilon\delta_{ij} = \sum_{i\in\mathcal{A}} \bar{b}_{ij} = \bar{q}_j\,.$$

Therefore, the change in $f_{\text{ncvx}}$ is given by

$$f_{\text{ncvx}}(\bar{\mathbf{b}} + \varepsilon\boldsymbol{\delta}, \bar{\mathbf{q}}) - f_{\text{ncvx}}(\bar{\mathbf{b}}, \bar{\mathbf{q}}) = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} \varepsilon\, w_i\, \delta_{ij} \log v_{ij} - \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} \varepsilon\, w_i\, \delta_{ij} \log \bar{q}_j = \varepsilon\, h(\boldsymbol{\delta}, \bar{\mathbf{q}})\,.$$

Note that $h(\boldsymbol{\delta}, \bar{\mathbf{q}})$ is a linear function in $\boldsymbol{\delta}$. So, if $h(\boldsymbol{\delta}, \bar{\mathbf{q}}) > 0$, then setting $\varepsilon = \max_x b_{i_{x+1} j_x}$ ensures that $f_{\text{ncvx}}(\bar{\mathbf{b}}+\varepsilon\boldsymbol{\delta}, \bar{\mathbf{q}}) \geq f_{\text{ncvx}}(\bar{\mathbf{b}}, \bar{\mathbf{q}})$, and $\bar{\mathbf{b}}+\varepsilon\boldsymbol{\delta} \in \mathcal{P}(\mathcal{A},\mathcal{G})$. In addition, the number of cycles in $G_{\text{supp}}(\bar{\mathbf{b}}+\varepsilon\boldsymbol{\delta})$ is strictly less than the number of cycles in $G_{\text{supp}}(\bar{\mathbf{b}})$.

Similarly, if $h(\boldsymbol{\delta}, \bar{\mathbf{q}}) \leq 0$, then setting $\varepsilon = -\max_x b_{i_x j_x}$ gives the same guarantees. Iterating this cycle cancelling process until the support does not contain any cycles leads to the required solution. ∎

By combining Lemma 1.2 with Lemma 1.3, we obtain the following corollary.

COROLLARY 2.1. *Let $\overline{\mathbf{b}}$ be any feasible point in $\mathcal{P}(\mathcal{A},\mathcal{G})$. Then there exists an acyclic solution, $\mathbf{b}^{\text{forest}}$, in the support of $\overline{\mathbf{b}}$ such that*

$$f_{\text{cvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{cvx}}(\overline{\mathbf{b}}) - D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u})\,.$$

*Moreover, such a $\mathbf{b}^{\text{forest}}$ can be found in time polynomial in $|\mathcal{A}|$ and $|\mathcal{G}|$.*

It is possible to save the additive factor of $D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u})$ in the above corollary by directly finding a first-order optimal solution to the non-convex program. This would lead to an approximation factor that improves on the current best $O(n \cdot w_{\max})$ [GHL+23]. Unfortunately, we can only find such a solution in pseudopolynomial time where the running time depends on the unary representation of both the weights $\mathbf{w}$ and the valuations $\mathbf{v}$. More details on this approach are provided in the full version of the paper.

Before presenting Algorithm 2, we give the proof of Theorem 1.1, which now follows directly from Theorem 1.2 and Corollary 2.1, as outlined below.

*Proof of Theorem 1.1.* Let $(\overline{\mathbf{b}}, \overline{\mathbf{q}})$ and $(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$, denote the feasible points defined in Steps 1 and 3 of Algorithm 1, respectively. Let $\sigma^\star$ be the assignment returned by Algorithm 2 on input $(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$. By Theorem 1.2, we have

$$\begin{aligned}
\text{NSW}(\sigma^\star) &\geq f_{\text{cvx}}(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}}) - D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - 2\log 2 - \frac{1}{2e} \\
&\overset{(i)}{\geq} f_{\text{cvx}}(\overline{\mathbf{b}}, \overline{\mathbf{q}}) - 2\cdot D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - 2\log 2 - \frac{1}{2e} \\
&\overset{(ii)}{\geq} \text{OPT} - 2\cdot D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - 2\log 2 - \frac{1}{2e}\,.
\end{aligned}$$

Here, $(ii)$ follows from Corollary 2.1, and $(iii)$ follows from Lemma 1.1. ∎

**2.1 Rounding an Acyclic Solution** Given an acyclic solution $\mathbf{b}$, Algorithm 2 returns an assignment with value comparable to $f_{\text{cvx}}(\mathbf{b})$, as stated in Theorem 1.2.

**Algorithm 2:** Algorithm for Rounding an Acyclic Solution

**Input**. NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, acyclic solution $(\mathbf{b}, \mathbf{q}) \in \mathcal{P}(\mathcal{A}, \mathcal{G})$

$(\mathbf{b}^\star, \mathbf{q}^\star) \leftarrow$ optimal solution of (CVX-Weighted) restricted to the support of $(\mathbf{b}, \mathbf{q})$

$F^\star \leftarrow G_{\text{supp}}(\mathbf{b}^\star)$ with every tree rooted at an agent node

Remove edges between item $j$ and its children in $F^\star$ whenever $q_j^\star < 1/2$ to get forest $\widetilde{F}$ **(Pruning)**

$L_i^\star \leftarrow$ set of leaf children of agent $i$ in $\widetilde{F}$ and let $L^\star = \cup_i \{L_i^\star\}$

$M^\star \leftarrow$ matching between $\mathcal{A} \to \mathcal{G} \backslash L^\star$ in $\widetilde{F}$ which maximizes weight function $w_{\widetilde{F}}(M) := \sum_{i \in \mathcal{A}} w_i \log \left( v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right)$ [1]

$\sigma^\star \leftarrow$ assignment of $\mathcal{G}$ to $\mathcal{A}$ with $\sigma^\star(j) = i$ if $j \in \{L_i^\star \cup M^\star(i)\}$ **(Matching)**

**Output.** $\sigma^\star$

Given an acyclic solution $\mathbf{b}$, Algorithm 2 first finds an optimal solution, denoted by $\mathbf{b}^\star$ to (CVX-Weighted) restricted to the support of $\mathbf{b}$, i.e., $\mathbf{b}^\star$ is the optimal solution to (CVX-Weighted) with input $\mathcal{A}, \mathcal{G}, \widetilde{\mathbf{v}}, \mathbf{w})$, where $\widetilde{v}_{ij} = 0$ if $b_{ij} = 0$, and $\widetilde{v}_{ij} = v_{ij}$ otherwise. This is crucial to later utilize the stability properties of stationary points of (CVX-Weighted).

Next, the algorithm implements a "pruning" step by sparsifying $\mathbf{b}^\star$ by removing edges between any item with $q_j^\star < 1/2$ and its children in $G_{\text{supp}}(\mathbf{b}^\star)$. This is equivalent to assigning item $j$ to its parent agent. As a result, any item with $q_j^\star < 1/2$ is a leaf in this pruned forest. Since removing edges will exclude certain items from being assigned to some agents, pruning can lead to a sub-optimal solution. We bound this loss in objective by showing the existence of a fractional solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ that is feasible in $\widetilde{F}$ after pruning, and has an objective comparable to the objective of $(\mathbf{b}^\star, \mathbf{q}^\star)$. For concrete details, see Section 3.

It is important to emphasize that the algorithm does not need to find such a solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$. The mere existence of $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ is enough to guarantee that the assignment returned by the algorithm will be good, as we explain below.

After the pruning step, the algorithm assigns every leaf item in the pruned forest to its parent. We use $L_i^\star$ to denote the set of leaf items whose parent is agent $i$ and $L^\star = \cup_{i \in \mathcal{A}} L_i^\star$ to denote the set of all leaf items in the pruned forest. So, each agent $i$ receives all the items in the bundle $L_i^\star$. In the matching step, the algorithm assigns at most one additional item to each agent by finding a maximum weight matching between agents $\mathcal{A}$ and items $\mathcal{G} \backslash L^\star$ (the set of non-leaf items in the pruned forest). This matching is determined using an augmented weight function, denoted by $w_{\widetilde{F}}$. The weight of a matching $M$ between $\mathcal{A}$ and $\mathcal{G} \setminus L^\star$ in the pruned forest is defined as follows:

$$w_{\widetilde{F}}(M) := \sum_{i \in \mathcal{A}} w_i \log \left( v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right),$$

where $v_{iM(i)} = 0$ if $i$ is not matched in $M$. Observe that this weight function exactly captures the weighted Nash Social Welfare objective when agent $i$ is assigned the item set $S_i := \{M(i) \cup L_i^\star\}$ for each $i \in \mathcal{A}$. Moreover, finding the optimal matching $M$ can be easily formulated as a maximum weight matching problem in a bipartite graph.

Since the standard linear programming relaxation for the bipartite matching problem is integral, it is enough to demonstrate the existence of a *fractional matching* with a large weight $w_{\widetilde{F}}$ in the pruned forest. In Section 3.2, we show how to construct a fractional matching corresponding to $\mathbf{b}^{\text{pruned}}$, such that the objective for the matching problem can be compared to the objective $f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{b}^{\text{pruned}})$. We emphasize that this matching corresponding to $\mathbf{b}^{\text{pruned}}$ is only required for the sake of analysis: to lower bound the performance of the matching returned by the algorithm. We do not require $\mathbf{b}^{\text{pruned}}$ for the execution of the algorithm.

[1] If agent $i$ in unmatched in $M$, we let $v_{iM(i)} = 0$

## 3 Rounding via the Non-Convex Relaxation

In this section, we prove Theorem 1.2 by establishing some properties of the set of support restricted optimal solutions of (CVX-Weighted). First, in Lemma 3.1, we show that any optimum whose support is restricted to a forest can be "pruned" to a feasible solution while only losing a constant factor in objective. Specifically, we show that given a support restricted optimum $(\mathbf{b}^\star, \mathbf{q}^\star)$, we can construct a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that any item with $q_j^{\text{pruned}} < 1/2$ is a leaf in support graph of $\boldsymbol{b}^{\text{pruned}}$, and $f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \geq f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - \log 2$.

Second, in Lemma 3.2, we demonstrate the existence of a matching in support graph of $\mathbf{b}^{\text{pruned}}$ such that the augment weight function of this matching differs from $f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}})$ by a constant factor. After presenting these two lemmas, we provide the proof of Theorem 1.2.

LEMMA 3.1. *Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be the optimal solution of (CVX-Weighted) in the support of some acyclic feasible point $\mathbf{b}^{\text{forest}}$. Let $F$ be the directed forest formed by $G_{\text{supp}}(\mathbf{b}^\star)$ when every tree is rooted at an agent node. Then, there exists an acyclic feasible point $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $G_{\text{supp}}(\mathbf{b}^\star)$ and*

- $q_j^{\text{pruned}} \geq q_j^\star$ *for any $j$ with $q_j^\star \geq 1/2$*
- *each item with $q_j^\star < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ connected to its parent in $F$,*
- $f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \geq f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - \log 2$.

The proof of Lemma 3.1 relies on the stability properties of first-order stationary solutions, as outlined in Section 3.1.

LEMMA 3.2. *Let $(\mathbf{b}, \mathbf{q})$ be an acyclic solution in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that every item with $q_j < 1/2$ is a leaf in $G_{\text{support}}(\mathbf{b})$. Let $S : \mathcal{A} \to 2^{\mathcal{G}}$ be a function such that for each agent $i$, $S(i)$ is a subset of leaf items connected to agent $i$ in $G_{\text{supp}}(\mathbf{b})$, and $S(i)$ contains all children of agent $i$ with $q_j < 1/2$. Then, there exists a matching $M$ in the subgraph of $G_{\text{supp}}(\mathbf{b})$ with vertices $\mathcal{A} \cup \{\mathcal{G} \setminus \cup_i \{S(i)\}\}$ such that*

$$\sum_{i \in \mathcal{A}} w_i \log \left( v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \geq f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) - \log 2 - \frac{1}{2e},$$

*where $v_{iM(i)} = 0$ if agent $i$ is not matched in $M$.*

We prove this lemma in Section 3.2.

*Proof of Theorem 1.2.* Given $(\mathbf{b}, \mathbf{q})$ such that $G_{\text{supp}}(\mathbf{b})$ is a forest, let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be the optimal solution of (CVX-Weighted) restricted to support of $\mathbf{b}$, let $\widetilde{F}$ denote the forest obtained after pruning $G_{\text{supp}}(\mathbf{b}^\star)$. Let $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ be a feasible solution guaranteed by Lemma 3.1 on input $(\mathbf{b}^\star, \mathbf{q}^\star)$.

Since $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subset of $G_{\text{supp}}(\mathbf{b}^\star)$, and every item with $q_j^\star < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$, we conclude that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $\widetilde{F}$. Furthermore, for any agent $i$, the set of leaf children of $i$ in $\widetilde{F}$ is a subset of the leaf children of $i$ in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$. To observe this, note that for any item $j$ with $q_j^\star < 1/2$, $j$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ only connected to its parent in $G_{\text{supp}}(\mathbf{b}^\star)$. For any item $j$ in $\widetilde{F}$ with $q_j^\star > 1/2$, we have $q_j^{\text{pruned}} > q_j^\star$. If $j$ is leaf in $\widetilde{F}$ connected to agent $i$, $q_j^{\text{pruned}} \geq q_j^\star$ and $G_{\text{supp}}(\mathbf{b}^\star) \subseteq \widetilde{F}$ enforces that $j$ is a leaf in $G_{\text{supp}}(\mathbf{b}^\star)$ connected to agent $i$.

Therefore, for each agent $i$, the set $L_i^\star$ is a subset of the set of leaves of agent $i$ in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ and $L_i^\star$ contains all the items with $q_j^{\text{pruned}} < 1/2$ in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$. So, the function $S(i) = L_i^\star$ satisfies the constraints of Lemma 3.2 with input $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$.

Using Lemma 3.2 on $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ with function $S(i) = L_i^\star$, we conclude that there exists a matching, $M$, in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ such that

$$\sum_{i \in \mathcal{A}} w_i \log \left( v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right) = \sum_{i \in \mathcal{A}} w_i \log \left( v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \geq f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e}.$$

Since $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $\widetilde{F}$, the matching $M$ is also a present in $\widetilde{F}$. Therefore, the matching $M^\star$ (and corresponding assignment $\sigma^\star$) returned by Algorithm 2 satisfies

$$\begin{aligned}
\text{NSW}(\sigma^\star) &= \sum_{i\in\mathcal{A}} w_i \log\left(v_{iM^\star(i)} + \sum_{j\in L_i^\star} v_{ij}\right) \\
&\overset{(i)}{\geq} \sum_{i\in\mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j\in L_i^\star} v_{ij}\right) \\
&\overset{(ii)}{\geq} f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e} \\
&\overset{(iii)}{\geq} f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - D_{\text{KL}}(\boldsymbol{w}\,||\,\boldsymbol{u}) - \log 2 - \frac{1}{2e} \\
&\overset{(iv)}{\geq} f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - D_{\text{KL}}(\boldsymbol{w}\,||\,\boldsymbol{u}) - 2\log 2 - \frac{1}{2e} \\
&\overset{(v)}{\geq} f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) - D_{\text{KL}}(\boldsymbol{w}\,||\,\boldsymbol{u}) - 2\log 2 - \frac{1}{2e}\,.
\end{aligned}$$

Here, $(i)$ follows from the optimality of $M^\star$, $(ii)$ follows from Lemma 3.2 and $(iii)$ follows from Lemma 1.3, $(iv)$ follows from Lemma 3.2, and $(v)$ follows from optimality of $\mathbf{b}^\star$. □

**3.1 Pruning Small Items** In this section, we prove Lemma 3.1 by establishing some properties of the set of first-order stationary solutions of (CVX-Weighted) in Lemma 3.3 and Lemma 3.4.

First, we show that any such stationary solution of (CVX-Weighted) is relatively stable, i.e., the change in function value when moving away from the stationary solution can be quantified in terms of how much we deviate from that solution. We formalize the stability of a first-order stationary solution of (1) as follows.

LEMMA 3.3. *Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be the optimal solution of (CVX-Weighted) in the support of some acyclic feasible point $\mathbf{b}^{\text{forest}}$. Let $(\mathbf{b}, \mathbf{q})$ be a feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that the support of $\mathbf{b}$ is a subset of the support of $\mathbf{b}^\star$, and for any $j \in \mathcal{G}$, if $q_j^\star = 1$, then $q_j = 1$. Then*

$$f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) = \sum_{j\in\mathcal{G}} \sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\frac{\sum_{i\in\mathcal{A}} w_i\, b_{ij}}{\sum_{i\in\mathcal{A}} w_i\, b_{ij}^\star}\right).$$

We provide the proof of this lemma in Appendix A.

Second, in Lemma 3.4, we show that any acyclic first-order stationary point of (CVX-Weighted) can be pruned to a feasible solution, denoted by $\mathbf{b}^{\text{pruned}}$, which is amenable to rounding. Specifically, we show that given a first-order stationary point $(\mathbf{b}^\star, \mathbf{q}^\star)$, we can construct a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that any item with $q_j^{\text{pruned}} < 1/2$ is a leaf in support of $\mathbf{b}^{\text{pruned}}$ and $b_{ij}^{\text{pruned}} \leq \min\{1, 2b_{ij}^\star\}$ for any agent $i$ and item $j$.

LEMMA 3.4. *Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be any acyclic feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$. Let $F$ be the directed forest formed by $G_{\text{supp}}(\mathbf{b}^\star)$ when every tree is rooted at an agent node. There exists a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ of $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $G_{\text{supp}}(\mathbf{b}^*)$,*

- *$q_j^\star \leq q_j^{\text{pruned}}$ for each item $j$ with $q_j^\star \geq 1/2$,*
- *each item with $q_j^\star < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ connected to its parent in $F$, and*
- *for any $(i, j) \in \mathcal{A} \times \mathcal{G}$, $b_{ij}^{\text{pruned}} \leq \min\{1, 2 \cdot b_{ij}^\star\}$.*

Before proving Lemma 3.4, we use Lemma 3.4 along with Lemma 3.3 to prove Lemma 3.1.

*Proof of Lemma 3.1.* By Lemma 3.4, there exists a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $G_{\text{supp}}(\mathbf{b})$ and $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ satisfies the first two items claimed in the lemma. Furthermore, for any $(i, j) \in \mathcal{A} \times \mathcal{G}$, $b_{ij}^{\text{pruned}} \leq \min\{1, 2 \cdot b_{ij}^\star\}$.

So using Lemma 3.3, the difference in objective between $(\mathbf{b}^\star, \mathbf{q}^\star)$ to $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ is bounded as follows

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) = \sum_{j\in\mathcal{G}} \sum_{i\in\mathcal{A}} w_i\, b_{ij}^{\text{pruned}} \log\left(\frac{\sum_{i\in\mathcal{A}} w_i\, b_{ij}^{\text{pruned}}}{\sum_{i\in\mathcal{A}} w_i\, b_{ij}^\star}\right).$$

Since $b_{ij}^{\text{pruned}} \le \min\{1,\, 2\, b_{ij}^\star\}$ for each $(i,j)$, we have $\sum_i w_i\, b_{ij}^{\text{pruned}} \le 2 \sum_i w_i\, b_{ij}^\star$.

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \le \sum_{j\in\mathcal{G}} \sum_{i\in\mathcal{A}} w_i\, b_{ij}^{\text{pruned}} \log 2\,. \tag{3.5}$$

The feasibility of $\mathbf{b}^{\text{pruned}}$ implies

$$\sum_{j\in\mathcal{G}} \sum_{i\in\mathcal{A}} w_i\, b_{ij}^{\text{pruned}} = \sum_{i\in\mathcal{A}} w_i \sum_{j\in\mathcal{G}} b_{ij}^{\text{pruned}} = \sum_{i\in\mathcal{A}} w_i = 1\,.$$

Plugging this bound in equation (3.5) completes the proof. □

Before proving Lemma 3.4, we need the following lemma about the feasibility of a solution when we decrease the $b_{ij}$ for some edge $(j \to i)$ in the support forest of $\mathbf{b}$.

LEMMA 3.5. *Let $(\mathbf{b}, \mathbf{q})$ be a feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that $G_{\text{supp}}(\mathbf{b})$ is a forest. Let $F$ be the directed forest formed by $G_{\text{supp}}(\mathbf{b})$ when every tree is rooted at an agent node. For a non-root agent $i$ in $F$, let item $j$ be its parent. Then for any $0 \le \delta \le \min\{b_{ij},\, 1 - b_{ij}\}$, there exists a feasible solution, $(\mathbf{b}^\delta, \mathbf{q}^\delta)$ such that $b_{ij}^\delta = b_{ij} - \delta$, $q_j^\delta = q_j - \delta$, $q_{j'}^\delta \ge q_{j'}$ for all $j' \in \mathcal{G}\backslash\{j\}$, and*

$$b_{i'j'}^\delta \begin{cases} \le \min\{1,\, 2b_{i'j'}\} & \text{if } i', j' \in T(i) \\ = b_{i'j'} & \text{otherwise}\,, \end{cases}$$

*where $T(x)$ denotes the subtree in the forrest $F$ rooted at $x$.*

*Proof of Lemma 3.4.* We will iteratively build the solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ satisfying these properties while ensuring that it remains feasible. For a vertex $x \in \mathcal{A} \cup \mathcal{G}$, let $\text{par}(x)$ denote its in parent in $G_{\text{supp}}(\mathbf{b}^\star)$, let $C(x)$ to denote the set of its children in $G_{\text{supp}}(\mathbf{b}^\star)$, and let $T(x)$ denote the sub-tree rooted at vertex $x$ in $G_{\text{supp}}(\mathbf{b}^\star)$.

Consider an item $j$ with $q_j^\star < 1/2$. To make the vertex corresponding to $j$ a leaf, the algorithm removes all the edges between item $j$ and its children $C(j)$. To reflect this change, we will update the solution $b^\star$ to an intermediate solution $\widetilde{b}$ such that the support of $\widetilde{b}$ does not contain any edges between item $j$ and its children. To maintain feasibility, we require:

$$\begin{aligned} \widetilde{q}_j = \widetilde{b}_{\text{par}(j)j} &= b^\star_{\text{par}(j)j} \\ \widetilde{b}_{ij} &= 0 \text{ for all } i \in C(j) \end{aligned} \tag{3.6}$$

Note that $q_j^\star < 1/2$ implies $b_{ij}^\star \le 1/2$. As a result, $b_{ij}^\star \le \min\{b_{ij}^\star, 1 - b_{ij}^\star\}$ for each $i \in C(j)$.

By iteratively applying Lemma 3.5 to edge $(j \to i)$ with $\delta = b_{ij}$ for each $i \in C(j)$, we arrive at the required solution $(\widetilde{b}, \widetilde{q})$ such that $\widetilde{b}_{ij} = 0$ for each $i \in C(j)$ and $\widetilde{q}_j = q_j - \sum_{i\in C(j)} b_{ij} = b_{\text{par}(j)j}$. Since $T(j)$ is the disjoint union of the sub-trees rooted at nodes in $C(j)$, for distinct $i_1, i_2 \in C(j)$, updating the sub-tree for $i_1$ does not affect the $b$ values for any edge in $T(i_2)$ and vice versa. Therefore, we have $q_{j'}^\star \le \widetilde{q}_{j'}$ for any item $j' \in T(j)$ and $\widetilde{b}_{i'j'} \le \min\{1, 2b^\star_{i'j'}\}$ for any $i', j' \in T(j)$.

Since we want to ensure that all items with $q_j^\star < 1/2$ become leaves, we must repeat the above process for any such item. The following fact is crucial to bound the values after multiple pruning processes: Pruning an item $j$ only changes $b$ values for edges in $T(j)$, and item $j$ becomes a leaf after that. So, if we prune ancestors of $j$ after pruning $j$, the $b$ values of edges in $T(j)$ are not changed.

So let $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ be the solution obtained by pruning the set of items $J = \{j \in \mathcal{G} : q_j^\star < 1/2\}$ in decreasing order of their height[2]. Note that pruning item $j$ does not decrease $q$ value for any item other than $j$.

[2]Note that pruning items in decreasing order of their height is only an artifact of the analysis. The algorithm can prune items with $q_j^\star < 1/2$ in any order.

Therefore, if $q_j^{\text{pruned}} < 1/2$, then item $j$ has already been pruned and is a leaf. For any item $j$ with $q_j^\star \geq 1/2$, its $q$ value only increases when its nearest ancestor is pruned, and this is the only time its $q$ value changes, we concludes that $q_j^{\text{pruned}} \geq q_j^\star$. This proves the second claim of the lemma.

To establish the third point of the lemma, observe that the $b$ variable for any edge in $G_{\text{supp}}(\mathbf{b}^\star)$ changes at most twice during the pruning process: If $q_j^\star \geq 1/2$, then item $j$ itself is not pruned, and $b$ values of edges incident to $j$ may change only when the nearest ancestor of $j$ is pruned. By Lemma 3.5, $b_{ij}^{\text{pruned}} \leq \min\{1,\ 2b_{ij}^\star\}$ for each $i \in \mathcal{A}$. If $q_j^\star < 1/2$, the $b$ value of any edge from $j$ to its children becomes zero when $j$ is pruned, and the $b$ value of the edge $(\text{par}(j) \to j)$ does not change when we prune $j$, and it may increase if $j$ has an ancestor that is also pruned after $j$. If so, we have $b_{\text{par}(j)j}^{\text{pruned}} \leq \min\{1,\ 2b_{\text{par}(j)j}^\star\}$.

□

**3.2 Fractional Matching and Analysis** In this section, we prove Lemma 3.2, which completes the proof of Theorem 1.2.

We establish Lemma 3.2 by proving two inequalities (in Lemmas 3.6 and 3.6) about the properties $f_{\text{ncvx}}$ for any feasible point whose support is a forest. Lemma 3.6 shows that $f_{\text{ncvx}}$ can be upper bounded by a linear function in $\mathbf{b}$ while losing a constant factor.

LEMMA 3.6. *Let $(\mathbf{b}, \mathbf{q})$ be an acyclic solution in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that every item with $q_j < 1/2$ is a leaf in $G_{\text{support}}(\mathbf{b})$. Let $S : \mathcal{A} \to 2^{\mathcal{G}}$ be a function such that for each agent $i$, $S(i)$ is a subset of leaf items connected to agent $i$ in $G_{\text{supp}}(\mathbf{b})$, and $S(i)$ contains all children of agent $i$ with $q_j < 1/2$. Then*

$$\sum_{i \in \mathcal{A}} w_i \left( \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left( \sum_{j \in S(i)} v_{ij} \right) \right) \geq f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) - \log 2 - \frac{1}{2e} .$$

Lemma 3.7 demonstrates how the linear function obtained from Lemma 3.7 can be used to as a lower bound for the maximum weight matching. A crucial component of this Lemma is the fact that any feasible $\mathbf{b}$ in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ corresponds to a point the matching polytope where all agents are matched.

LEMMA 3.7. *Let $(\mathbf{b}, \mathbf{q})$ be an acyclic solution in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that every item with $q_j < 1/2$ is a leaf in $G_{\text{support}}(\mathbf{b})$. Let $S : \mathcal{A} \to 2^{\mathcal{G}}$ be a function such that for each agent $i$, $S(i)$ is a subset of leaf items connected to agent $i$ in $G_{\text{supp}}(\mathbf{b})$, and $S(i)$ contains all children of agent $i$ with $q_j < 1/2$. Then, there exists a matching $M$ in the subgraph of $G_{\text{supp}}(\mathbf{b})$ with vertices $\mathcal{A} \cup \{\mathcal{G} \setminus \cup_i \{S(i)\}\}$ such that*

$$\sum_{i \in \mathcal{A}} w_i \log \left( v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \geq \sum_{i \in \mathcal{A}} w_i \left( \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left( \sum_{j \in S(i)} v_{ij} \right) \right) , \tag{3.7}$$

*where $v_{iM(i)} = 0$ if agent $i$ is not matched in $M$.*

Lemma 3.6 and Lemma 3.6 together establish Lemma 3.2. In the rest of this section, we provides the proofs of Lemma 3.6 and Lemma 3.7.

*Proof of Lemma 3.6.* Recall that

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) &= \sum_{i \in \mathcal{A}} w_i \sum_{j \in \mathcal{G}} b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} w_i \sum_{j \in \mathcal{G}} b_{ij} \log q_j \\ &= \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log q_j + \sum_{i \in \mathcal{A}} w_i \left( \sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} \right) . \end{aligned} \tag{3.8}$$

where the last equation follows from the fact that every item in $L_i$ is a leaf, i.e., there is exactly one agent ...

Therefore, for any $j \notin L$, $\widetilde{q}_j \geq 1/2$. As a result,

$$- \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log q_j \leq \log 2 \sum_{i \in \mathcal{A}} w_i \, b_{ij} . \tag{3.9}$$

Plugging this bound into equation (3.8) gives

$$f_{\text{ncvx}}(\mathbf{b},\mathbf{q}) \leq \sum_{i\in\mathcal{A}} w_i \sum_{j\notin S(i)} b_{ij}\log v_{ij} - \sum_{i\in\mathcal{A}} w_i \sum_{j\notin S(i)} b_{ij}\log 2 + \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\in S(i)} b_{ij}\log v_{ij} - b_{ij}\log b_{ij}\right). \tag{3.10}$$

As $\mathbf{b}\in\mathcal{P}(\mathcal{A},\mathcal{G})$, we have $\sum_{j\notin S(i)} b_{ij} = 1 - \sum_{j\in S(i)} b_{ij}$ for every agent $i$. Substituting this in equation (3.10) yields

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b},\mathbf{q}) &\leq \sum_{i\in\mathcal{A}} w_i \sum_{j\notin S(i)} b_{ij}\log v_{ij} + \sum_{i\in\mathcal{A}} w_i\log 2 + \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\in S(i)} b_{ij}\log v_{ij} - b_{ij}\log b_{ij} - b_{ij}\log 2\right) \\ &= \sum_{i\in\mathcal{A}} w_i \sum_{j\notin S(i)} b_{ij}\log v_{ij} + \log 2 + \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\in S(i)} b_{ij}\log v_{ij} - b_{ij}\log b_{ij} - b_{ij}\log 2\right), \end{aligned} \tag{3.11}$$

where the last equation follows from $\sum_i w_i = 1$.

For each agent $i\in\mathcal{A}$, Claim 1.1 implies that

$$\sum_{j\in S(i)} b_{ij}\log v_{ij} - b_{ij}\log b_{ij} \leq \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right) - \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} b_{ij}\right).$$

So, for any agent $i$,

$$\begin{aligned} \sum_{j\in S(i)} b_{ij}\log v_{ij} - b_{ij}\log b_{ij} - b_{ij}\log 2 &\leq \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right) - \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} b_{ij}\right) - \sum_{j\in S(i)} b_{ij}\log 2 \\ &\leq \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right) + \frac{1}{2e}, \end{aligned} \tag{3.12}$$

where the last inequality follows from $-x\log(x) - x\log 2 \leq 1/(2e)$ for all $x\geq 0$ applied to $x = \sum_{j\in S(i)} b_{ij}$.

Substituting (3.12) in (3.11), we get

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b},\mathbf{q}) &\leq \sum_{i\in\mathcal{A}} w_i \sum_{j\notin S(i)} b_{ij}\log v_{ij} + \log 2 + \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right) + \frac{1}{2e}\right) \\ &= \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\notin S(i)} b_{ij}\log v_{ij} + \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right)\right) + \log 2 + \frac{1}{2e}, \end{aligned}$$

where the last inequality follows from $\sum_{i\in\mathcal{A}} w_i = 1$. □

*Proof of Lemma 3.7.* In this proof, we will analyze a matching that either assigns the bundle $S(i)$ to an agent or a single item $j\notin\cup_i S(i)$. Observe that the matching $M$ clearly finds an assignment with a larger objective as

$$\log\left(v_{iM(i)} + \sum_{j\in S(i)} v_{ij}\right) \geq \max\left\{\log v_{iM(i)}, \log\left(\sum_{j\in S(i)} v_{ij}\right)\right\}.$$

So, for each agent $i\in\mathcal{A}$, we create a new leaf item $\ell_i$ with $v_{i\ell_i} = \sum_{j\in S(i)} v_{ij}$ corresponding to the sey of items in $S(i)$. Define $S := \cup_i S(i)$ and $\widetilde{\mathcal{G}} := \{\mathcal{G}\backslash S\}\cup\{\ell_i\}_{i\in\mathcal{A}}$. We show that the maximum weight matching in the bipartite graph $(\mathcal{A},\widetilde{\mathcal{G}})$ suffices to prove the lemma. As the matching polytope is integral, to show there exists a matching of large objective, it is enough to demonstrate a fractional matching of large value.

Using $\mathbf{b}$, we define fractional assignment variables $\mathbf{x}$ as follows:

$$x_{ij} := b_{ij} \quad \forall i \in \mathcal{A}, j \in \{\mathcal{G} \backslash L\}$$
$$x_{i\ell_i} := \sum_{j \in S(i)} b_{ij} \quad \forall i$$

Using $\mathbf{x}$, we can re-write the L.H.S. of equation (3.7) as

$$\sum_{i \in \mathcal{A}} w_i \left( \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left( \sum_{j \in S(i)} v_{ij} \right) \right) = \sum_{i \in \mathcal{A}} \sum_{j \in \widetilde{\mathcal{G}}} x_{ij}\, w_i \log v_{ij} \,. \tag{3.13}$$

Observe that $\mathbf{x}$ lies in the convex hull of matchings between agents $\mathcal{A}$ and items $\widetilde{\mathcal{G}}$ in which every agent is matched as $\mathbf{x}$ satisfies the following properties:

$$\sum_{j \in \widetilde{\mathcal{G}}} x_{ij} = \sum_{j \notin S(i)} b_{ij} + \sum_{j \in S(i)} b_{ij} = 1 \quad \forall i \in \mathcal{A}$$
$$\sum_{i \in \mathcal{A}} x_{ij} \leq 1 \quad \forall j \in \widetilde{\mathcal{G}} \,.$$

Here, the last inequality for item $j \notin S$ is inherited from the feasibility of $\mathbf{b}$. The constraint for $\ell_{i'}$ for some $i' \in \mathcal{A}$ is implied by the constraint $\sum_{i \in \mathcal{A}} x_{ij} = x_{i'j} = \sum_{j \in S(i)} b_{ij} \leq \sum_{j \in \mathcal{G}} b_{ij} \leq 1$, where the last constraint follows from feasibility of $\mathbf{b}$.

Using the integrality of the matching polytope, there exists a matching $\widetilde{M} : \mathcal{A} \to \widetilde{\mathcal{G}}$ such that

$$\sum_{i \in \mathcal{A}} \sum_{j \in \widetilde{\mathcal{G}}} x_{ij}\, w_i \log v_{ij} \leq \sum_{i \in \mathcal{A}} w_i \, \log v_{i\widetilde{M}(i)} \,. \tag{3.14}$$

Now consider a $M : \mathcal{A} \to \mathcal{G}$ with $M(i) = \emptyset$ if $\widetilde{M}(i) = \ell_i$, and $M(i) = \widetilde{M}(i)$ otherwise. Then

$$\sum_{i \in \mathcal{A}} w_i \, \log v_{i\widetilde{M}(i)} \leq \sum_{i \in \mathcal{A}} w_i \, \log \left( v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \,. \tag{3.15}$$

Then equations (3.13), (3.14) and (3.15) together imply

$$\sum_{i \in \mathcal{A}} w_i \, \log \left( v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \geq \sum_{i \in \mathcal{A}} w_i \left( \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left( \sum_{j \in S(i)} v_{ij} \right) \right) \,.$$

□

## 4 Conclusion and Open Questions

In this paper, we introduced a convex and a non-convex relaxation for the weighted (asymmetric) Nash Social Welfare problem. Both of these relaxations play a crucial role in obtaining the approximation algorithm for the problem. There are two natural open questions. First, is the factor $\exp\left(D_{\mathrm{KL}}(\boldsymbol{w} \,||\, \boldsymbol{u})\right)$ necessary in the approximation guarantee? Equivalently, is it possible to obtain a constant factor approximation for the weighted Nash Social Welfare problem? It is important to emphasize that we lose the $\exp\left(D_{\mathrm{KL}}(\boldsymbol{w} \,||\, \boldsymbol{u})\right)$ when relating the objectives of the two relaxations; we only lose a constant factor when rounding the non-convex relaxation. There may exist a direct approach to approximately solve the non-convex formulation that gives an improved approximation guarantee.

The second question is whether the techniques introduced in this work generalize to more general valuation functions, in particular, submodular valuations for the weighted Nash Social Welfare problem. While there are constant factor approximation algorithms for symmetric Nash Social Welfare with submodular valuations, obtaining anything better than $O(nw_{\max})$-approximation for the weighted variant of the problem remains an open question.

## References


[AGSS17] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash Social Welfare, Matrix Permanent, and Stable Polynomials. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:12, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[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 *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2274–2290. SIAM, 2018.

[BCE+16] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D.Editors Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[BKV18] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 557–574, 2018.

[BT96] Steven J. Brams and Alan D. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996.

[BT05] Julius B. Barbanel and Alan D. Taylor. *The Geometry of Efficient Fair Division*. Cambridge University Press, 2005.

[CDG+17] Richard Cole, Nikhil Devanur, Vasilis Gkatzelis, Kamal Jain, Tung Mai, Vijay V Vazirani, and Sadra Yazdanbod. Convex program duality, fisher markets, and nash social welfare. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 459–460, 2017.

[CG15] Richard Cole and Vasilis Gkatzelis. Approximating the nash social welfare with indivisible items. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 371–380, 2015.

[CKM+19] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32, 2019.

[CM04] Suchan Chae and Hervé Moulin. Bargaining among groups: An axiomatic viewpoint. *International Journal of Game Theory*, 39, 02 2004.

[FKL12] Hu Fu, Robert Kleinberg, and Ron Lavi. Conditional equilibrium outcomes via ascending price processes with applications to combinatorial auctions with item bidding. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, page 586, New York, NY, USA, 2012. Association for Computing Machinery.

[GHL+23] Jugal Garg, Edin Husić, Wenzheng Li, László A Végh, and Jan Vondrák. Approximating nash social welfare by matching and local search. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1298–1310, 2023.

[GHV21] Jugal Garg, Edin Husić, and László A Végh. Approximating nash social welfare under rado valuations. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1412–1425, 2021.

[GKK20] Jugal Garg, Pooja Kulkarni, and Rucha Kulkarni. Approximating nash social welfare under submodular valuations through (un) matchings. In *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms*, pages 2673–2687. SIAM, 2020.

[HLZ13] Harold Houba, Gerard Laan, and Yuyu Zeng. Asymmetric nash solutions in the river sharing problem. *Strateg. Behav. Environ.*, 4, 03 2013.

[HS72] John C. Harsanyi and Reinhard Selten. A generalized nash solution for two-person bargaining games with incomplete information. *Management Science*, 18(5):P80–P106, 1972.

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

[KN79] Mamoru Kaneko and Kenjiro Nakamura. The nash social welfare function. *Econometrica: Journal of the Econometric Society*, pages 423–435, 1979.

[Lee17] Euiwoong Lee. Apx-hardness of maximizing nash social welfare with indivisible items. *Information Processing Letters*, 122:17–20, 2017.

[LV07] Annick Laruelle and Federico Valenciano. Bargaining in committees as an extension of nash's bargaining theory. *Journal of Economic Theory*, 132(1):291–305, 2007.

[LV22] Wenzheng Li and Jan Vondrák. A constant-factor approximation algorithm for nash social welfare with submodular valuations. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2022.

[NJ50] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the econometric society*, pages 155–162, 1950.

[NNRR14] Nhan-Tam Nguyen, Trung Thanh Nguyen, Magnus Roos, and Jörg Rothe. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Autonomous agents and multi-agent systems*, 28(2):256–289, 2014.

[Rot15] Jrg Rothe. *Economics and Computation: An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*. Springer Publishing Company, Incorporated, 1st edition, 2015.
[RW98] J. Robertson and W. Webb. *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998.
[YIWZ17] S. Yu, E. C. 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(5):709–729, October 2017.
[You94] H. Peyton Young. *Equity: In Theory and Practice*. Princeton University Press, 1994.

## A Omitted Proofs and Lemmas

*Example.* Consider a weight vector $\mathbf{w}$ with $w_1 = \frac{1}{\log n}$ and $w_2 = \ldots = w_n = \frac{1}{n-1}\left(1 - \frac{1}{\log n}\right)$. Then

$$\begin{aligned} D_{\mathrm{KL}}(\mathbf{w} \,||\, \mathbf{u}) &= \frac{1}{\log n} \log\left(\frac{n}{\log n}\right) + \left(1 - \frac{1}{\log n}\right) \log\left(\frac{n}{n-1}\left(1 - \frac{1}{\log n}\right)\right) \\ &= 1 - \frac{\log\log n}{\log n} + \left(1 - \frac{1}{\log n}\right) \log\left(\frac{n}{n-1}\right) + \left(1 - \frac{1}{\log n}\right) \log\left(1 - \frac{1}{\log n}\right) \\ &\leq 1 + \log\left(\frac{n}{n-1}\right) \leq 2\,. \end{aligned}$$

□

*Proof of Lemma 1.1.* Let $\sigma : \mathcal{G} \to \mathcal{A}$ be the optimal assignment of an instance of Nash Social Welfare. For each agent $i \in \mathcal{A}$, define $V_i = \sum_{j \in \sigma^{-1}(i)} v_{ij}$. Using $\sigma$, we define a vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ as

$$b_{ij} := \begin{cases} \frac{v_{ij}}{V_i} & \text{if } \sigma(j) = i \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that $\sum_{i \in \mathcal{A}} b_{ij} \leq 1$ for each $j \in \mathcal{G}$ and $\sum_{i \in \mathcal{G}} b_{ij} = 1$ for each $i \in \mathcal{A}$. We will now show that $f_{\mathrm{cvx}}(\mathbf{b})$ and $f_{\mathrm{ncvx}}(\mathbf{b})$ are both equal to $\mathrm{NSW}(\sigma)$.

$$\begin{aligned} f_{\mathrm{cvx}}(\mathbf{b}) &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log v_{\sigma(j)j} - \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log\left(\frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i \\ &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log\left(\frac{V_{\sigma(j)}}{w_{\sigma(j)}}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i \\ &= \sum_{i \in \mathcal{A}} w_i \sum_{j \in \sigma^{-1}(i)} \frac{v_{ij}}{V_i} \log\left(\frac{V_i}{w_i}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i \\ &= \sum_{i \in \mathcal{A}} w_i \log\left(\frac{V_i}{w_i}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i = \sum_{i \in \mathcal{A}} w_i \log V_i = \mathrm{NSW}(\sigma)\,, \text{and} \\ f_{\mathrm{ncvx}}(\mathbf{b}) &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log v_{\sigma(j)j} - \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log\left(\frac{v_{\sigma(j)j}}{V_{\sigma(j)}}\right) \\ &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log V_{\sigma(j)} = \sum_{i \in \mathcal{A}} w_i \sum_{j \in \sigma^{-1}(i)} \frac{v_{ij}}{V_i} \log V_i \\ &= \sum_{i \in \mathcal{A}} w_i \log V_i = \mathrm{NSW}(\sigma)\,. \end{aligned}$$

For the second claim in the lemma, when $w_i = 1/n$ for each $i$, for any $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, we have

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}) &= \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log v_{ij} - \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log\left(\frac{\sum_{i\in\mathcal{A}} b_{ij}}{n}\right) - \log n \\ &= \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log v_{ij} - \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log\left(\sum_{i\in\mathcal{A}} b_{ij}\right) + \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log n - \log n \\ &= \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log v_{ij} - \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log\left(\sum_{i\in\mathcal{A}} b_{ij}\right), \end{aligned}$$

where we used $\sum_j b_{ij} = 1$ for every $i$ in the last inequality. Simply substituting $w_i = 1/n$ for each $i$ in $f_{\text{ncvx}}$ completes the proof. □

*Proof of Lemma 1.3.* We will show that

$$f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) = D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - D_{\text{KL}}(\mu\,||\,\theta)\,,$$

where $\mu, \theta$ are two probability distributions on $\mathcal{G}$ given by

$$\mu(j) = \sum_{i\in\mathcal{A}} w_i\, b_{ij} \quad \text{and} \quad \theta(j) = \frac{\sum_{i\in\mathcal{A}} b_{ij}}{n}\,.$$

Using $\sum_{i\in\mathcal{A}} w_i = 1$ and $\sum_{j\in\mathcal{G}} b_{ij} = 1$ for each $i \in \mathcal{A}$, one can verify that $\sum_{j\in\mathcal{G}} \mu(j) = 1 = \sum_{j\in\mathcal{G}} \theta(j)$.

Expanding the difference between the functions gives

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) &= \sum_{i\in\mathcal{A}} w_i\log w_i - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij}\log\left(\sum_{i\in\mathcal{A}} w_i\, b_{ij}\right) + + \sum_{i,j} w_i\, b_{ij}\log\left(\sum_{i\in\mathcal{A}} b_{ij}\right) \\ &= \sum_{i\in\mathcal{A}} w_i\log w_i - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij}\log\left(\frac{\sum_{i\in\mathcal{A}} w_i\, b_{ij}}{\sum_{i\in\mathcal{A}} b_{ij}}\right) \\ &= \sum_{i\in\mathcal{A}} w_i\log w_i + \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij}\log n - \sum_{j\in\mathcal{G}} \mu(j)\log\left(\frac{\mu(j)}{\theta(j)}\right) \\ \text{(using } \textstyle\sum_j b_{ij} = 1\text{)} \qquad &= \sum_{i\in\mathcal{A}} w_i\log(n w_i) - \sum_{j\in\mathcal{G}} \mu(j)\log\left(\frac{\mu(j)}{\theta(j)}\right) \\ &= D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - D_{\text{KL}}(\mu\,||\,\theta)\,. \end{aligned}$$

As $D_{KL}(\mu, \theta) \geq 0$, the above equation implies

$$f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) \leq D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u})\,.$$

For the lower bound it suffices to show that $D_{\text{KL}}(\mu\,||\,\theta) \leq D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u})$. To see this, we can expand the

definition:

$$
\begin{aligned}
D_{\mathrm{KL}}(\mu \,||\, \theta) &= \sum_{j\in\mathcal{G}} \left(\sum_{i\in\mathcal{A}} w_i b_{ij}\right) \log\left(\frac{n\sum_{i\in\mathcal{A}} w_i b_{ij}}{\sum_{i\in\mathcal{A}} b_{ij}}\right) \\
&= \log(n) + \sum_{j\in\mathcal{G}} \left(\sum_{i\in\mathcal{A}} w_i b_{ij}\right) \log\left(\frac{\sum_{i\in\mathcal{A}} w_i b_{ij}}{\sum_{i\in\mathcal{A}} b_{ij}}\right) \\
&= \log(n) + \sum_{j\in\mathcal{G}} \left(\sum_{i\in A} b_{ij}\right) \left(\sum_{i\in\mathcal{A}} \frac{b_{ij}}{\sum_{i\in A} b_{ij}} \cdot w_i\right) \log\left(\sum_{i\in\mathcal{A}} \frac{b_{ij}}{\sum_{i\in A} b_{ij}} \cdot w_i\right) \\
&\le \log(n) + \sum_{j\in\mathcal{G}} \left(\sum_{i\in A} b_{ij}\right) \left(\sum_{i\in\mathcal{A}} \frac{b_{ij}}{\sum_{i\in A} b_{ij}}\right) w_i \log(w_i) \\
&= \log(n) + \sum_{j\in\mathcal{A}} w_i \log(w_i) \sum_{j\in\mathcal{G}} b_{ij} \\
&= \log(n) + \sum_{i\in\mathcal{A}} w_i \log(w_i) = D_{\mathrm{KL}}(\mathbf{w} \,||\, \mathbf{u})\,.
\end{aligned}
$$

Here, the only inequality is the convexity of $x\log(x)$, and the last equality follows from the feasibility of $\mathbf{b}$. □

As a first step to establish the stability of a stationary point of (1), we need the following the property about its support.

LEMMA A.1. *Let $\mathbf{b}^\star$ be the optimal solution of (CVX-Weighted) (even with support restriction) with $q_j^\star = \sum_{i\in A} b_{ij}^\star$. Then there exist real numbers $\{\lambda_i\}_{i=1}^n$ and $\{\eta_j\}_{j=1}^m \ge 0$ such that $\eta_j(1-q_j^\star) = 0$ for all $j \in \mathcal{G}$ and if $b_{i,j}^\star > 0$, then*

$$
w_i \log v_{ij} = w_i \log\left(\sum_{i\in\mathcal{A}} w_i b_{ij}\right) + w_i + \lambda_i + \eta_j\,.
$$

*Proof of Lemma A.1.* If $\mathbf{b}^\star$ is a first-order stationary solution of 1, then using the KKT conditions, there exist $\lambda_i \in \mathbb{R}$ and $\alpha_{ij}, \eta_j \ge 0$ such that

$$
\frac{\partial L}{\partial b_{ij}^\star} = w_i \log v_{ij} - w_i - w_i \log\left(\sum_{i\in\mathcal{A}} w_i b_{ij}^\star\right) - \lambda_i - \eta_j + \alpha_{ij} = 0 \tag{A.1}
$$

and

$$
\begin{aligned}
\eta_j(1 - \sum_{i\in\mathcal{A}} b_{ij}^\star) &= 0 \\
\alpha_{ij} b_{ij}^\star &= 0\,.
\end{aligned}
\qquad \text{(Complementary slackness)}
$$

Using the complementary slackness condition, if $b_{ij}^\star > 0$, then

$$
w_i \log v_{ij} = w_i + w_i \log\left(\sum_{i\in\mathcal{A}} w_i b_{ij}^\star\right) + \lambda_i + \eta_j\,.
$$

□

*Proof.* Expanding the difference between the two function values, we get

$$
\begin{aligned}
f_{\mathrm{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\mathrm{cvx}}(\mathbf{b}, \mathbf{q}) = &\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} \left(b_{ij}^\star - b_{ij}\right) \cdot w_i \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij}^\star \log\left(\sum_{i\in\mathcal{A}} w_i\, b_{ij}^\star\right) \\
&+ \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i\in\mathcal{A}} w_i\, b_{ij}\right).
\end{aligned} \tag{A.2}
$$

Since $(\mathbf{b}^\star, \mathbf{q}^\star)$ is locally optimal, Lemma A.1 implies that there exist real numbers $\lambda_i$ for each $i \in \mathcal{A}$ and $\eta_j \geq 0$ for each $j \in \mathcal{G}$ such that $\eta_j(1 - q_j^\star) = 0$ for all $j \in \mathcal{G}$, and if $b_{ij}^\star > 0$, then

$$w_i \log v_{ij} = w_i \log \left( \sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star \right) + \lambda_i + w_i + \eta_j \, .$$

Substituting this value of $v_{ij}$ in equation (A.2) gives

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) &= \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} (b_{ij}^\star - b_{ij}) \left( w_i \log \left( \sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star \right) + \lambda_i + w_i + \eta_j \right) \\ &\quad - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star \log \left( \sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star \right) + \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left( \sum_{i \in \mathcal{A}} w_i \, b_{ij} \right) \\ &= \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left( \frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}}{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star} \right) + \sum_{i \in \mathcal{A}} (\lambda_i + w_i) \left( \sum_{j \in \mathcal{G}} b_{ij}^\star - \sum_{j \in \mathcal{G}} b_{ij} \right) \\ &\quad + \sum_{j \in \mathcal{G}} \eta_j \left( \sum_{i \in \mathcal{A}} b_{ij}^\star - \sum_{i \in \mathcal{A}} b_{ij} \right) \end{aligned}$$

Using $\sum_{j \in \mathcal{G}} b_{ij} = \sum_{j \in \mathcal{G}} b_{ij}^\star = 1$ for every $i \in \mathcal{A}$, we get

$$f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left( \frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}}{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star} \right) + \sum_{j \in \mathcal{G}} \eta_j \left( q_j^\star - q_j \right) \, ,$$

where the last equation follows from the definitions of $q_j$ and $q_j^\star$.

Note that by Lemma A.1, $\eta_j(1 - q_j^\star) = 0$ for any $j \in \mathcal{G}$. So if $q_j^\star < 1$, then $\eta_j = 0$ and therefore $\eta_j(q_j^\star - q_j) = 0$. If $q_j^\star = 1$, then by the hypothesis of the Lemma, $q_j = 1$, and again we obtain that $\eta_j(q_j^\star - q_j) = 0$. Using this bound in the above equation gives

$$f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left( \frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}}{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star} \right) .$$

□

*Proof of Lemma 3.5.* For $x \in \mathcal{A} \cup \mathcal{G}$, let $C(x)$ denote the children of node $x$ in $F$ and let $T(x)$ denote the sub-tree rooted at node $x$. We will prove this lemma by induction of the height of agent $i$ by building $(\mathbf{b}^\delta, \mathbf{q}^\delta) \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ inductively.

For the base case, assume agent $i$ has height 1, i.e., $T(i)$ consists of only leaf item nodes that are the children of node $i$. Note that setting $b_{ij}^\delta = b_{ij} - \delta$ and $q_j^\delta = q_j - \delta$ only violates the Agent constraint for agent $i$. So we only need to change the values of $b$ in $T(i)$ to make the solution feasible.

By the feasibility of $b$, $b_{ij} + \sum_{k \in C(i)} b_{ik} = 1$ and for every item node $k \in C(i)$, $q_k = b_{ik} < 1$. Using Lemma A.2 with $\alpha = b_{ij}$ and $\beta_k = b_{ik}$, there exists some $\delta_k \in \mathbb{R}_{\geq 0}$ for each $k \in C(i)$ such that

$$\begin{aligned} b_{ij} - \delta + \sum_{k \in C(i)} b_{ik}(1 + \delta_k) &= 1 \\ b_{ik}(1 + \delta_k) &\leq 1 \quad \forall k \in C(i) \\ 0 \leq \delta_k &\leq 1 \quad \forall k \in C(i) \, . \end{aligned}$$

So, for each $k \in C(i)$, we set $q_k^\delta = b_{ik}^\delta = b_{ik}(1 + \delta_k)$. As every item in $C(i)$ is a leaf, this gives $q_k^\delta \leq 1$ for all $k \in C(i)$, and

$$\sum_{k \in C(i)} b_{ik}^\delta = b_{ij} - \delta + \sum_{k \in C(i)} b_{ik}(1 + \delta_k) = 1 \, .$$

Therefore $(b^\delta, q^\delta)$ is feasible. Additionally, for every $k \in C(i)$, $q^\delta_k = q_k(1+\delta_k) \le 2q_k$ as $\delta_k \le 1$.

For the induction hypothesis, assume that the lemma is true whenever the height of agent is at most $\ell - 1$ for some integer $\ell \ge 1$. We will show that the statement also holds when the height of agent $i$ is $\ell$.

Again, that setting $b^\delta_{ij} = b_{ij} - \delta$ and $q^\delta_j = q_j - \delta$ violates the Agent constraint for agent $i$. Similar to the base case, we can find $\delta_k \in (0,1)$ for each $k \in C(i)$ such that $b_{ik}(1+\delta_k) \le 1$ and

$$b_{ij} - \delta + \sum_{k \in C(i)} b_{ik}(1+\delta_k) = 1\,.$$

Setting $b^\delta_{ik} = b_{ik}(1+\delta_k)$ for each $k \in C(i)$ will ensure that $b^\delta$ satisfies the Agent constraint for agent $i$. However, this can violate the Item constraint for some item $k \in C(i)$, as $q^\delta_k = q_k + \delta_k b_{ik}$. So we inductively update the values of $b$ and $q$ for the sub-tree rooted at item $j$ for which such a violation occurs.

Consider an item $k \in C(i)$ such that $q^\delta_k = q_k + \delta_k b_{ik} > 1$. Define $\gamma := q_k + \delta_k b_{ik} - 1$. Since $q_k \le 1$, we have $\gamma \le \delta_k b_{ik} \le b_{ik}$ as $\delta_k \le 1$. We will decrease $b_k$ to ensure that $q^\delta_k$ is at most 1. Using the fact that $q_k = \sum_{i' \in C(k)} b_{i'k} + b_{ik}$, we can further bound $\gamma$ as follows.

$$\begin{aligned} \gamma = q_k + \delta_k b_{ik} - 1 &= \sum_{i' \in C(k)} b_{i'k} + b_{ik} + \delta_k b_{ik} - 1 \\ \text{(using } b_{ik}(1+\delta_k) \le 1\text{)} \qquad &\le \sum_{i' \in C(k)} b_{i'k} \end{aligned}$$

Therefore, there exist numbers $\gamma_{i'} \ge 0$ for each $i' \in C(k)$ such that $\gamma_{i'} \le b_{i'k}$ and $\sum_{i' \in C(k)} \gamma_{i'} = \gamma$.

We set $b^\delta_{i'k} = b_{i'k} - \gamma_{i'}$ for each $i' \in C(k)$. Then $q^\delta_k = b^\delta_{ik} + \sum_{i' \in C(k)} b^\delta_{i'k}$ satisfies the following inequalities.

$$\begin{aligned} q^\delta_k &= b^\delta_{ik} + \sum_{i' \in C(k)} b^\delta_{i'k} \\ &= b_{ik}(1+\delta_k) + \sum_{i' \in C(k)} (b_{i'k} - \gamma_{i'}) = 1 + \gamma - \sum_{i' \in C(k)} \gamma_{i'} = 1 \ge q_j \quad \text{and} \\ q^\delta_k &= b_{ik}(1+\delta_k) + \sum_{i' \in C(k)} (b_{i'k} - \gamma_{i'}) < b_{ik}(1+\delta_k) + \sum_{i' \in C(k)} b_{i'k} \\ &= q_k + \delta_k b_{ik} < 2q_k\,, \end{aligned}$$

where we used $\delta_k \le 1$ and $b_{ik} \le q_k$ for the last inequality.

Furthermore, as $\gamma_{i'} \le \gamma \le b_{ik}$, we have $\gamma_{i'} \le q_k - b_{i'k} \le 1 - b_{i'k}$ for any $i' \in C(k)$. So for each $i' \in C(k)$,

$$\gamma_{i'} \le \min\{b_{i'k},\ 1 - b_{i'k}\}\,.$$

Using the induction hypothesis, for each $i' \in C(k)$, there exists feasible $(b^{\gamma_{i'}}, q^{\gamma_{i'}})$ which differs from $(b,q)$ only in the sub-tree rooted at $i'$ such that for any item $j' \in T(i')$, $q_{j'} \le q^{\gamma_{i'}}_{j'} \le \min\{1, 2q_{j'}\}$.

For item $j' \in T(i')$, setting $q^\delta_{j'} = q^{\gamma_{i'}}_{j'}$ completes the proof.4 □

LEMMA A.2. *Let $\alpha > 0$ and $\beta_1, \dots, \beta_k > 0$ with $\alpha + \sum_{j=1}^k \beta_i = 1$. For any $0 < \delta \le \min\{\alpha, 1-\alpha\}$, there exist real numbers $(\delta_1, \dots, \delta_k)$ such that*

$$\begin{aligned} \alpha - \delta + \sum_{j \in [k]} \beta_j(1+\delta_j) &= 1 \\ \beta_j(1+\delta_j) &\le 1 \quad \forall j \in [k] \\ 0 \le \delta_j &\le 1 \quad \forall j \in [k] \end{aligned} \tag{A.3}$$

*Proof.* As the above system contains only linear constraints on $\delta_j$, we use Farkas' Lemma to show the existence

of $\{\delta_j\}_{j=1}^k$. Re-arranging the constraints gives

$$\sum_{j\in[k]} \beta_j\delta_j = \delta \tag{A.4}$$
$$\beta_j\delta_j \le 1-\beta_j \quad \forall j\in[k]$$
$$0 \le \delta_j \le 1 \quad \forall j\in[k]$$

If there do not exist real numbers $\{\delta_j\}_{j=1}^k$ satisfying (A.4), then by Farkas' Lemma, there exist real numbers $\eta, \{\gamma_j\}_{j=1}^k, \{\lambda_j\}_{j=1}^k$ such that

$$\beta_j\eta + \beta_j\gamma_j + \lambda_j \ge 0 \quad \forall j\in[k] \tag{A.5}$$
$$\gamma_j, \lambda_j \ge 0$$
$$\delta\eta + \sum_{j\in[k]}(1-\beta_j)\gamma_j + \sum_{j\in[k]}\lambda_j < 0 \tag{A.6}$$

Adding equation (A.5) for all $j\in[k]$, we get

$$\eta\sum_{j\in[k]}\beta_j + \sum_{j\in[k]}\beta_j\gamma_j + \sum_{j\in[k]}\lambda_j \ge 0\,.$$

Since $\alpha + \sum_{j\in[k]}\beta_j = 1$, this implies $\eta(1-\alpha) + \sum_{j\in[k]}\beta_j\gamma_j + \sum_{j\in[k]}\lambda_j \ge 0$. In addition, since $\beta_i > 0$, we also have $\alpha < 1$. Therefore dividing by $1-\alpha$ gives

$$\sum_{j\in[k]}\frac{\beta_j\gamma_j}{1-\alpha} + \sum_{j\in[k]}\frac{\lambda_j}{1-\alpha} \ge -\eta\,. \tag{A.7}$$

On the other hand, equation (A.6) implies

$$-\eta > \sum_{j\in[k]}\frac{(1-\beta_j)\gamma_j}{\delta} + \sum_{j\in[k]}\frac{\lambda_j}{\delta} \tag{A.8}$$

On combining equations (A.7) and (A.8), we obtain

$$\sum_{j\in[k]}\frac{\beta_j\gamma_j}{1-\alpha} + \sum_{j\in[k]}\frac{\lambda_j}{1-\alpha} > \sum_{j\in[k]}\frac{(1-\beta_j)\gamma_j}{\delta} + \sum_{j\in[k]}\frac{\lambda_j}{\delta}\,. \tag{A.9}$$

We will now derive a contradiction to (A.9).

As $\delta \le 1-\alpha$ , we have $1/(1-\alpha) \le 1/\delta$, we have

$$\sum_{j\in[k]}\frac{\lambda_j}{1-\alpha} \le \sum_{j\in[k]}\frac{\lambda_j}{\delta}\,, \tag{A.10}$$

where we also use the fact that $\lambda_j > 0$ for all $j\in[k]$.

In addition, any $j\in[k]$

$$\frac{\beta_j}{1-\alpha} - \frac{(1-\beta_j)}{\delta} \le \frac{\beta_j}{1-\alpha} - \frac{(1-\beta_j)}{\alpha} = \frac{\alpha+\beta_j-1}{\alpha(1-\alpha)} \le 0\,. \tag{A.11}$$

Here, the first inequality follows from $\delta \le a$ and the last inequality follows from the fact that $\alpha + \sum_{j\in[k]}\beta_j = 1$ and $\alpha, \beta_j > 0$.

On adding equation (A.10) with equation (A.11) for all $j\in[k]$, we obtain

$$\sum_{j\in[k]}\frac{\beta_j\gamma_j}{1-\alpha} + \sum_{j\in[k]}\frac{\lambda_j}{1-\alpha} \le \sum_{j\in[k]}\frac{(1-\beta_j)\gamma_j}{\delta} + \sum_{j\in[k]}\frac{\lambda_j}{\delta}\,,$$

which contradicts (A.9). Therefore, there exist real numbers $\{\delta_j\}_{j=1}^k$ satisfying (A.3). ∎