Department of Computer Science
George Mason University
Technical Report Series

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/
703-993-1530

Long-term Fairness with Bounded Worst-case Losses

Gabriel Balan
gbalan@cs.gmu.edu

Dana Richards
richards@cs.gmu.edu

Sean Luke
sean@cs.gmu.edu

Technical Report GMU-CS-TR-2008-2

Abstract

How does one repeatedly choose actions so as to be fairest to the multiple beneficiaries of those actions? We examine approaches to discovering sequences of actions for which the worst-off beneficiaries are treated maximally well, then secondarily the second-worst-off, and so on. We formulate the problem for the situation where the sequence of action choices continues forever; this problem may be reduced to a set of linear programs. We then extend the problem to situations where the game ends at some unknown finite time in the future. We demonstrate that an optimal solution is NP-hard, and present two good approximation algorithms.

1 Introduction

Consider the problem of repeatedly assigning two AI professors to teach two classes offered by their department each semester. One class is much harder than the other one, so during any single semester any one-to-one assignment is unfair to one of the professors. One fair solution would be for the professors to teach both classes together. But assuming this requires more overall effort than teaching the classes separately, this solution would be inefficient over the long run.

There is of course a better solution. If the two professors instead took turns teaching the hard class, then in the long run their average utilities would be *more fair* than in the one-to-one assignments and *more efficient* than in the sharing assignment. This is the rough idea behind *long-term fairness*: repeated interactions offer opportunities for improved efficiency and fairness over the single interaction scenario. But in general the solutions will not be as simple as alternating assignments (e.g. suppose we extended the previous example with multiple assignments, involving many professors and classes).

The research presented here examines the following framework: there are a number of *beneficiaries* (e.g. professors), which receive different *rewards* from each of a finite set of *actions* (e.g. class assignments). The actions are chosen with replacement, and actions chosen early do not restrict what actions can be chosen later, or their rewards. This framework, borrowed from [18], is very similar to the repeated normal-form game framework from game theory,[1] except there is a single *decision maker* that chooses actions for the good of all beneficiaries.

We are ultimately in the game-theoretic multiple decision maker case. But it turns out that the single decision maker situation has been inadequately studied, and so our focus here is on that case.

We define a beneficiary's *utility* as the average of all rewards he received in the past. We will use the term *utility profile* to refer to the vector of utilities (one utility per beneficiary) that they derive from past actions.

In order to compare utility profiles, we use a fairness concept called *leximin*. Leximin is a total-order relation defined as follows: a utility profile U_1 is preferable to another utility profile U_2 if the beneficiary with the minimum utility in U_1 is better-off than the minimum utility beneficiary in U_2; ties are broken by comparing the utilities of the the second-worst beneficiary in each utility profile, then third-worst, and so on. From a procedural point of view, one should sort in increasing order the two utility profiles, and compare the sorted versions lexicographically. We chose leximin for our work because it is widely used in the literature, it is well understood, and it

[1] The example assigning professors to classes is a variant of the game "Battle of the Sexes" (also known as "Bach or Stravinsky"), without the miscoordinated joint-actions.

incorporates a certain measure of efficiency (a leximin-optimal profile is also pareto-optimal), making it a good fit for long-term fairness.

In this paper we derive upper bounds for this notion of utility in infinitely repeated games and propose a particular approach to approximation solutions for more realistic cases (i.e. a finite sequence of actions).

2 Infinite-length Games

An action may give high rewards to some beneficiaries and low rewards to others, so sticking to just one action might be unfair to some beneficiaries, and thus leximin-undesirable. However, if beneficiaries receive different rewards from different actions we may be able to improve all beneficiaries' reward averages by playing a combination of actions.

One approach to playing combinations is to use a periodic, repeated sequence of actions. In previous literature [17, 18], distributed algorithms for discovering such sequences found suboptimal ones. We will show optimal solutions, albeit with non-distributed algorithms.

Because beneficiaries' utilities change with every new action being played, it is not obvious how to compare arbitrary sequences. Periodic sequences are thus convenient because the resulting utility profiles always converge and one can straightforwardly compare periodic sequences by comparing the utility profiles they converge to.

Periodic sequences are intuitively appealing, but even if one can find the periodic sequence with the best limit, that can still be suboptimal. We will show at the end of this section that in some problem instances with irrational coefficients there might exist infinite non-periodic sequences that achieve, at the limit, leximin-superior utility profiles to any utility profile achievable by a periodic sequence. Our algorithms are guaranteed to produce sequences converging to the optimal utility profile, and, whenever possible, those sequences are periodic.

In this section we will (1) prove that convergent sequences (periodic or non-periodic) are sufficient and (2) identify the class of sequences with the leximin-optimal limit-point. In the following sections we will propose additional requirements to impose on this class of sequences and then provide algorithms that produce sequences satisfying these requirements.

Problem 1. ***Base Problem*** *Let there be a set $\mathbb{A}$ of n_a actions affecting a set $\mathbb{B}$ of n_b beneficiaries through the reward function $R : \mathbb{A} \times \mathbb{B} \to \mathbb{R}$. Let $\mathbb{S}$ be the set of infinite sequences of actions from $\mathbb{A}$:*

$$\mathbb{S} = \{S = \langle s_1, s_2, \dots \rangle | \forall i \in \mathbb{N} : s_i \in \mathbb{A}\}.$$

Let $\mathbb{U}$ be the set of all possible utility profiles (vectors) achievable from following any sequence for any number of time steps:

$$\mathbb{U} = \{U \in \mathbb{R}^{n_b} | \exists t \in \mathbb{N}, \exists S \in \mathbb{S} : U_b = \frac{1}{t}\sum_{j=1}^{t} R(s_j, b)\}.$$

The goal of the problem is to find $U^\star = \sup(\mathbb{U})$, the supremum (least upper bound) with respect to leximin over the set of all achievable utility profiles.

In order to solve this problem we first show that it is enough to focus on a subset of "well-behaved" sequences, by proving that they achieve the entire set of possible utility profiles $\mathbb{U}$. We then provide a mapping of that subset into the n_a-dimensional simplex which can be searched efficiently using linear-programming-based algorithms from the literature.

To make things easier, we refer to the elements of $\mathbb{A}$ as $\{1 \dots n_a\}$ and the elements of $\mathbb{B}$ as $\{1 \dots n_b\}$. Let $\mathbb{S}' \subset \mathbb{S}$ be the set of all sequences S where the *proportions* of the n_a actions converge. Formally:

$$\mathbb{S}' = \{S \in \mathbb{S} | \forall j \in \mathbb{A} : \exists \lim_{t \to \infty} \frac{1}{t} k_j(S_t)\}$$

where S_t is the subsequence of S consisting of the first t elements and k is the *count* function (so $k_j(S_t)$ is equal to the number of times action j is used in the first t positions of sequence S). We denote with $F(S)$ the vector of action proportions (or fractions) for S (i.e. $F_j(S) = \lim_{t\to\infty} \frac{1}{t} k_j(S_t)$).

We use the notation $U(S_t)$ to mean the vector of utilities achieved after following the first t steps of sequence S (i.e. U's component for beneficiary b is $U_b(S_t) = \frac{1}{t}\sum_{i=1}^{t} R(s_j, b)$). Because the action proportions converge, the sequence of utility vectors $\langle U(S_1), U(S_2), \dots \rangle$ must also converge component by component and we denote its limit point by the vector $U(S)$:

$$U_b(S) = \lim_{t\to\infty} U_b(S_t) = \sum_{j=1}^{n_a} F_j(S) R(s_j, b) \qquad (1)$$

Also, let $\mathbb{U}'$ be the set of utility vectors achievable by sequences in $\mathbb{S}'$. Obviously $\mathbb{U}' \subseteq \mathbb{U}$ since $\mathbb{S}' \subset \mathbb{S}$. Also, $\mathbb{U}' \supseteq \mathbb{U}$ because $\forall U \in \mathbb{U}$ there must exist $S \in \mathbb{S}$ and $\tau \in \mathbb{N}$ such that $U = U(S_\tau)$, and based on S and τ it is trival to build a sequence $S' \in \mathbb{S}'$ that achieves U (e.g. $s'_t = s_{(t-1) \pmod{\tau}+1}$). Therefore $\mathbb{U}' = \mathbb{U}$.

Let $\mathbb{U}''$ be the set of limit points of $\mathbb{U}'$ ($\mathbb{U}'' = \{U | \exists S \in \mathbb{S}' : U = U(S)\}$). We claim that

$$\sup(\mathbb{U} = \mathbb{U}') = \sup(\mathbb{U}'') \qquad (2)$$

because all achievable utility vectors are also limit points. Formally, $\forall \tau \in \mathbb{N}$, we can build $S' \in \mathbb{S}'$ such that $U(S') = U(S_\tau)$ (again it is enough to chose $s'_t = s_{(t-1) \pmod{\tau}+1}$).

Because $U(S)$ depends only on $F(S)$, there is an obvious one-to-one correspondence between $\mathbb{U}''$ and the n_a-dimensional unit simplex.

Once reformulated as an optimization problem over a compact and convex set, the problem becomes readily solvable with the algorithm proposed in [12]. In our particular case, the algorithm consists of solving $O(n_a^2)$ linear programs (LPs), but approximate results based on a floating-point fixed-length representation might be required to make sure the complexity of this algorithm does not dominate that of the algorithms we propose here.

The algorithm produces two vectors, $U^\star$ and $F^\star$. $U^\star$ is the leximin-optimal utility vector, and $F^\star$ is a point inside the simplex such that:

$$\forall b \in \mathbb{B} : \sum_{i=1}^{n_a} R(b,i) \times F_i^\star = U_b^\star$$

$U^\star$ is unique [4], but $F^\star$ does not have to be. Since $U^\star$ is the leximin-maximal utility vector in $\mathbb{U}''$ (by the algorithm's guarantee), and it is unique, it implies that $U^\star = \sup(\mathbb{U}'')$, and, by Equation 2:

$$U^\star = \sup(\mathbb{U}). \tag{3}$$

We revisit the existence of a periodic sequence for a given optimal utility profile solution $U^\star$. If such a periodic policy exists, let c_j denote the number of times of action j appears in one period (we call c_j the *multiplicity* of action j). Then by normalizing the vector $c = [c_1 \dots c_{n_a}]$ one should obtain a valid $F^\star$ vector. If all rewards are rational then there must exist an $F^\star$ with rational components,[2] and hence a periodic sequence ($c_j = \frac{p_j}{q_j} \times P$, where $F_j^\star = \frac{p_j}{q_j}$, and where p_j, q_j are integers and P is the *least common multiple* for $q_1 \dots q_{n_a}$). However irrational rewards could mean there is no $F^\star$ with rational coefficients, and no periodic sequence.

We illustrate this point with a modified version of the professor example: both professors get a reward of 0 from teaching the hard class, but teaching the easy class gives one professor a reward of 1 and the other a reward of $\sqrt{2}$. Then $U^\star = [\frac{\sqrt{2}}{1+\sqrt{2}}, \frac{\sqrt{2}}{1+\sqrt{2}}]$ and there is a unique $F^\star = [\frac{\sqrt{2}}{1+\sqrt{2}}, \frac{1}{1+\sqrt{2}}]$. We make the observation that depending on the values in $F^\star$, the multiplicities could be arbitrarily large. Thus, one can see the non-periodic sequences due to $F^\star$ having (some) irrational components as special cases of periodic sequences where the period length is infinite.

3 Finite-length Games

Solving the **Base Problem** provides $U^\star$, the least upper bound over the set of achievable utility profiles. If the game is guaranteed to last forever, one could be satisfied with the goal of finding a sequence that achieves $U^\star$ at the limit. But in more practical applications, one must consider the implications of having the process end after a finite number of steps, or more generally, having the length of the sequence of actions drawn from some probability distribution.

Consider the professor-assigment example from the introduction: the first actions corresponds to the assignment where the first professor teaches the easy class, the second action corresponds to the assignment where the second professor teaches the easy class, and the third action is the assignment where they teach the classes together:

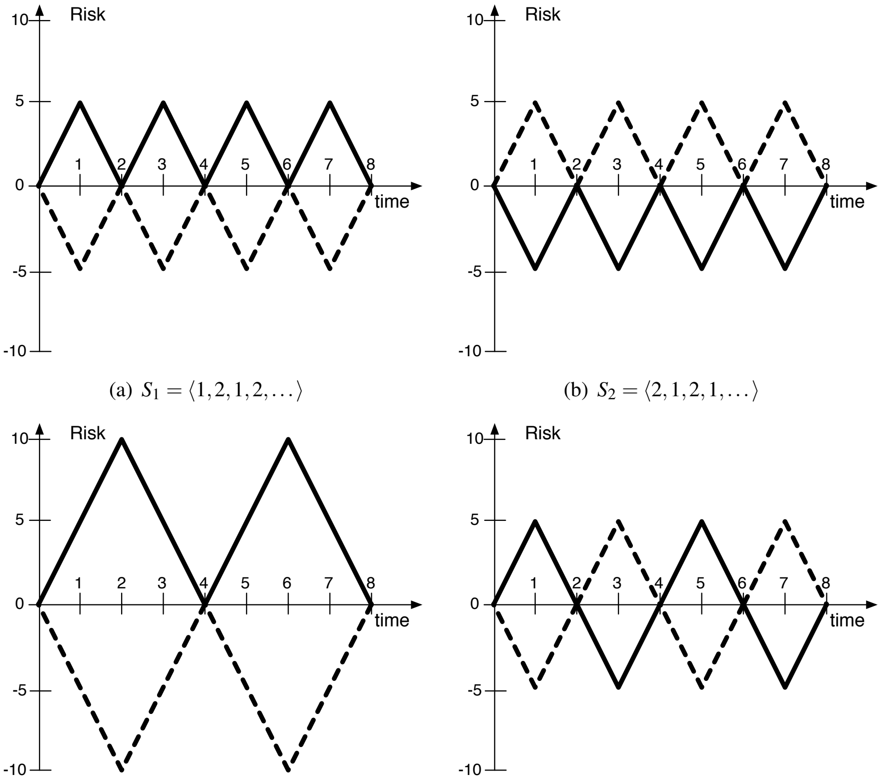
Example 1.

		Beneficiaries	
		b_1	b_2
	1	*10*	*0*
Actions	*2*	*0*	*10*
	3	*1*	*1*

The utility least upper bound for Example 1 is $U^\star = [5,5]$, which can only be achieved through $F^\star = [0.5, 0.5, 0]$, and which coincides with the optimal solution if the game lasts for an even number of steps: use actions 1 and 2 in equal proportions. However, one should use action 3 if the game lasts for only one round. This shows that the optimal sequence of decisions depends on the duration of the game, and so if the duration is not known in advance, some sort of tradeoff might be required.

One could chose an action stochastically, using the values in $F^\star$ as probabilities. This approach produces the leximin-optimal *expected rewards* (equal to $U^\star$). This sort of guarantee is usually referred to in the economics literature as *ex-ante* fairness (ex-ante means "beforehand" in latin). If all beneficiaries are risk-neutral, a mechanism choosing repeatedly from the $F^\star$ distribution promises leximin-optimal *expected utilities* before the process starts, but not necessarily leximin-optimal actual utilities when the game ends For Example 1 this approach would flip a coin between actions 1 and 2; although the expected rewards (and utilities) are equal to $U^\star$, if the game ends after two rounds, there is a 50% chance that one beneficiary accumulated 10 reward units and the other none. Moreover, if the game continues, the probability that the difference between the two will shrink is equal to the probability that the difference will grow. Thus, the weakness of this ex-ante approach would be its lack of concern with paying reparations for unfairness resulting from past randomness.

[2] The algorithm that computes $U^\star$ does only arithmetic operations (e.g. using Simplex to handle its linear programming calls), and since the input values (rewards) are rational, so must be the output values (i.e. $F^\star$).



(a) $S_1 = \langle 1,2,1,2,\dots\rangle$

(b) $S_2 = \langle 2,1,2,1,\dots\rangle$

(c) $S_3 = \langle 1,1,2,2,1,1,2,2,\dots\rangle$

(d) $S_4 = \langle 1,2,2,1,1,2,2,1,\dots\rangle$

Figure 1: Risks as functions of time for the two beneficiaries in Example 1 (the solid line for the first and the dotted line for the second) as produced by various periodic sequences.

One can address the weaknesses of the previous method with a deterministic approach, aiming to find a sequence which leximin-optimizes beneficiaries' *expected utilities* weighted by the probabilities in the distribution of game lengths. Unlike the previous method, this could use action 3 in Example 1, provided there is a high enough probability that the game will end after one step.

Both previous methods assume *risk-neutral* beneficiaries. Given that *leximin* is a *risk-averse* fairness concept ("no one is left behind"), it makes sense to instead focus on a *risk-averse* solution approach: minimize the largest amount a beneficiary risks losing due to the game ending prematurely. We define the *risk* of beneficiary k at time step t while executing sequence S as the difference between the rewards accumulated by beneficiary k during the first t steps, and the amount he was entitled to, which is $t \times U_k^\star$.

$$Risk_b(S,t) = \sum_{i=1}^{t} R(s_i,b) - t \times U_b^\star \tag{4}$$

We use the term *worst risk* (WR) of a sequence S to mean a lower bound on all risks values, regardless of beneficiary (i.e. $WR \leq Risk_b(S,t)$, $\forall t \in \mathbb{N}$ and $\forall b \in \mathbb{B}$). The worst risk is a negative values, and its absolute value is an upper bound on the largest amount that any beneficiary can lose.

This approach is not meant to replace expectation optimization; we are simply adding another tool to the arsenal available for this problem. It is reasonable to expect the computational complexity of the expectation-optimization approach to grow with the maximum game length, and so our proposed solution is particularly suitable to scenarios with very large game durations (our approach is insensitive to the distribution of game durations). Furthermore, lower-bounding the worst risk will always make the utilities converge to $U^\star$ as $t \longrightarrow \infty$:

$$U_b(S_t) = \frac{1}{t}\sum_{i=1}^{t} R(s_i,b) = \frac{Risk_b(S,t)}{t} + U_b^\star$$

$$U_b(S_t) - U_b^\star \geq \frac{\text{risk lower bound}}{t} \longrightarrow 0$$

Note that the risks of all beneficiaries cannot be simultaneously positive, unless they are all zero. Otherwise one can use the sequence of actions used up to that point to build an infinite sequence with a utility vector leximin-superior to the optimal vector $U^\star$. Thus there must exist strictly negative risks during the implementation of any non-empty sequence. The only exception would be if there is some action j such that $\forall b : R(j,b) = U_b^\star$, but that case is trivial: simply play action j at each step, with 0 risk for each beneficiary.

For convenience, we define for each action j a vector $X_j = [X_{j,1} \ldots X_{j,n_b}]$, where $X_{j,b} = R(j,b) - U_b^\star$, the amount action j changes the risk of beneficiary b. Note that:

$$Risk_b(S,t) = \sum_{i=1}^{t} [R(s_i,b) - U_b^\star] = \sum_{i=1}^{t} X_{s_i,b} \tag{5}$$

$$\forall b \in \mathbb{B} : \sum_{j=1}^{n_a} X_{j,b} \times F_j^\star = 0 \tag{6}$$

Let us illustrate the concept of risk using Example 1. The X vectors are as follows: $X_1 = [5,-5]$, $X_2 = [-5,5]$ and $X_3 = [-4,-4]$. We will compare the following periodic action sequences (see Figure 1).

$$\begin{aligned}
S_1 &= \langle 1,2,1,2,\ldots \rangle \\
S_2 &= \langle 2,1,2,1,\ldots \rangle \\
S_3 &= \langle 1,1,2,2,1,1,2,2,\ldots \rangle \\
S_4 &= \langle 1,2,2,1,1,2,2,1,\ldots \rangle
\end{aligned}$$

Sequence S_1 makes the first beneficiary's risks equal to 5 on odd steps and 0 on even steps, while the second beneficiary's risks are 0 on odd steps and -5 on even ones. Therefore the second beneficiary risks coming up 5 units short if the game stops after an odd number of steps and breaks even otherwise. Sequence S_2 has identical effects as S_1, but to different beneficiaries, so the two are equally good. Using sequence S_3, the second beneficiary risks losing as much as 10 units, so we prefer S_1 (or S_2) to S_3.

Note that using sequence S_1 makes the second beneficiary take all the "bad" (negative) risks, while the first beneficiary takes only "good" (positive) risks. Although sequence S_1 is as fair as possible with respect to average utilities, one cannot help but notice a "second degree" unfairness. Instead, one might consider the goal of optimizing cumulative risks (e.g. sequence S_4 alternates which beneficiary is taking negative risks). This leads to the following paradox: although intuitively one prefers S_4 to S_1 or S_2, the worst risks for all beneficiaries during sequences S_1 or S_2 (i.e. -5 and 0) are leximin superior to the worst risks during S_4 (i.e. -5 and -5). This suggests that one might consider optimizing only the "min" of the worst risks, instead of leximin optimizing all beneficiaries' worst risks.

Another possible refinement comes from the observation that not all beneficiaries get the same expected utility, so a rich beneficiary could afford risking to lose more than a poor one. Therefore, as long as $U^\star$ has strictly positive components, one can consider optimizing *worst relative risks* ($RelRisk_b(A,t) = Risk_b(A,t)/U_b^\star$) instead of optimizing the worst risks.

Problem 2. ***Worst Risk Maximization Problem*** *Given the setup in the* ***Base Problem****, and $K \in \mathbb{N}$ such that $K > |V|$, find a sequence of actions S of length K with a maximal* worst risk.

Theorem 1. *Problem 2 is NP-hard.*

Proof. We prove Problem 2 is NP-hard through a reduction from **Partition Problem** [7]. In the **Partition Problem** one is given a multiset V of integer numbers and has to decide if V can be partitioned into two subsets whose elements sum to the same value. We will show that the answer to this decision problem is "YES" if and only if the optimal worst risk for a corresponding **Worst Risk Maximization Problem** instance is equal to a specific value.

For an arbitrary **Partition Problem** instance $V = \{v_1, \ldots, v_{|V|}\}$ we construct the associated **Worst Risk Maximization Problem** instance as follows. The instance has $|V|+1$ actions and $|V|+3$ beneficiaries. We differentiate between two types of beneficiaries. The first $|V|+1$ beneficiaries, which we refer to as $1 \ldots |V|+1$, receive the following rewards:

$$R(i,j) = \begin{cases} \delta - \frac{\sigma}{2} & \text{if } i = j \\ \delta + \frac{\sigma}{2 \times |V|} & \text{otherwise} \end{cases}$$

where σ is the sum of elements in V (i.e. $\sigma = \sum_{i=1}^{|V|} v_i$), and δ is an arbitrary constant. The remaining two beneficiaries (which we refer to as α and β) get the following rewards:

$$R(i,\alpha) = \begin{cases} \delta + v_i & \text{if } i \neq |V|+1 \\ \delta - \sigma & \text{otherwise} \end{cases}$$

$$R(i,\beta) = \begin{cases} \delta - v_i & \text{if } i \neq |V|+1 \\ \delta + \sigma & \text{otherwise} \end{cases}$$

The computational effort for the transformation is obviously linear in the size of the input.

The first step is to prove that $U^\star = [\delta \ldots \delta]$. Note that each action dispenses the same sum of rewards: $(|V|+3) \times \delta$. It follows that the sum of all beneficiaries' utilities is $(|V|+3) \times \delta$ at any time, regardless of the sequence of actions. Therefore no vector U can be leximin superior to $[\delta \dots \delta]$ since U can't have a component strictly greater than δ without having some other component strictly smaller than δ. Note that the utility vector $[\delta \dots \delta]$ is achievable, e.g., $U(b, S_{|V|+1}) = \delta$ for every beneficiary b, if the sequence S starts with an arbitrary permutation of the $|V|+1$ actions. Since the vector $[\delta \dots \delta]$ is achievable and no utility profile can be leximin-superior, then $U^\star$ must be equal to $[\delta \dots \delta]$.

For any sequence S it holds that $Risk_{s_1}(S,1) = -\frac{\sigma}{2}$ (whichever action j is used first, there is a beneficiary j whose risk is $-\frac{\sigma}{2}$), so the worst risk value $WR \le -\frac{\sigma}{2}$.

Claim *The answer to the* ***Partition Problem*** *instance is "YES" (i.e. there exists $V' \subset V$ such that $\sum_{v\in V'} v = \sum_{v \in V-V'} v = \frac{\sigma}{2}$) if and only if the worst risk touches its upper bound (i.e. $WR = -\frac{\sigma}{2}$).*

First we will prove the "IF" part of the claim: if the **Worst Risk Maximization Problem** solver produces a sequence S such that $Risk_k(S,t) = -\frac{\sigma}{2}$, then V can be partitioned into two subsets of equal size.

We submit that each of the $|V|+1$ actions is used exactly once in the first $|V|+1$ positions of S. If action j were to show up k times ($k \ge 2$), then $Risk_j(S, |V|+1) = -\frac{\sigma}{2} \times k + (|V|+1-k) \times \frac{\sigma}{2\times|V|} = \frac{\sigma}{2}\left(-k + \frac{|V|+1-k}{2\times|V|}\right) \le \frac{\sigma}{2}\left(-2 + \frac{|V|-1}{2\times|V|}\right) = -\frac{\sigma}{2}\left(1 + \frac{|V|+1}{2\times|V|}\right) < -\frac{\sigma}{2}$, which is a contradiction. Also, if action j were not used at all during the first $|V|+1$ positions of S then some other action i had to be used more than once, leading to the same contradiction.

Given that each action is used exactly once in the first $|V|+1$ steps, $Risk_j(S,t) \ge -\frac{\sigma}{2}$ $\forall t \in \{1 \dots |V|+1\}$, $\forall j \in \{1 \dots |V|+1\}$, because $Risk_j(S,t)$ is the sum of at most one negative term (i.e. $-\frac{\sigma}{2}$) and several positive ones (i.e. $\frac{\sigma}{2\times|V|}$). As for the beneficiaries α and β, their worst risks depend on the subset V' of elements in S that go before the first occurrence of action $|V|+1$. Beneficiary β's risk decreases monotonically from zero until action $|V|+1$ is chosen, then abruptly becomes positive and thereafter decreases monotonically to zero. Thus the worst risk for beneficiary β occurs right before action $|V|+1$:

$$-\frac{\sigma}{2} \le Risk_\beta(S, |V'|) = -\sum_{v \in V'} v \Rightarrow \sum_{v\in V'} v \le \frac{\sigma}{2}$$

Beneficiary α's worst risk occurs immediately after action $|V|+1$, since $Risk_\alpha(S,t) = -Risk_\beta(S,t)$, $\forall S, \forall t$:

$$-\frac{\sigma}{2} \le Risk_\alpha(S, |V'|+1) = \sum_{v\in V'} v - S \Rightarrow \sum_{v \in V'} v \ge \frac{\sigma}{2}$$

Consequently $\sum_{v\in V'} v = \frac{\sigma}{2} \Rightarrow \sum_{v \in V-V'} v = \frac{\sigma}{2}$, concluding the proof for the first part of the claim.

We now prove the "ONLY-IF" part of the claim: if there exists a partition of V into V' and $V-V'$ such that $\sum_{v\in V'} v = \sum_{v\in V-V'} v$, then there exists a sequence of length K that generates a worst risk of $-\frac{\sigma}{2}$.

Let S be a periodic sequence whose period consists of an arbitrary permutation of the actions in $\{i | v_i \in V'\}$ followed by action $|V|+1$ and an arbitrary permutation of the actions in $\{i | v_i \in V - V'\}$. Because the sequence S is periodic, the sequence of risks produced by S is also periodic. Therefore it is enough to verify that all risks are greater than or equal to $-\frac{\sigma}{2}$ during the first $|V|+1$ time steps. Note that each action is used exactly once during the first $|V|+1$ steps, so we can verify the worst risk for each beneficiary using arguments similar to those in the first part of the claim's proof. There is a single action j that affects beneficiary j's risk negative way, so $Risk_j(S,t) \ge -\frac{\sigma}{2}$. Beneficiary α's risk improves monotonically from zero to $\frac{\sigma}{2}$ during the first $|V'|$ steps; then becomes $-\frac{\sigma}{2}$ at the next step as a result of action $|V|+1$; then improves monotonically afterwards. Beneficiary β's risk worsens monotonically, until it reaches $-\frac{\sigma}{2}$ after step $|V'|$; afterwards it becomes equal $\frac{\sigma}{2}$ due to action $|V|+1$ and will remain positive until the end of the period.

By building sequence S we have shown that a worst risk of $-\frac{\sigma}{2}$ is possible, and since $WR \le -\frac{\sigma}{2}$, then $WR = -\frac{\sigma}{2}$ is optimal. Therefore a correct **Worst Risk Maximization Problem** solver must find some sequence with the same performance. This concludes the proof for the second part of the claim.

Therefore one can decide the answer to any instance of the **Partition Problem** by solving the corresponding **Worst Risk Maximization Problem** instance. To summarize: compute the risks for the first $|V|+1$ steps of the optimal-WR sequence. If at any time some risk was strictly less than $-\frac{\sigma}{2}$, then the partitioning is impossible; otherwise make a multiset V' consisting of all v_j such that action j encountered before action $|V|+1$, and return the partition V', $V-V'$.

The problem instance transformation and result interpretation are obviously polynomial in the size of the input, so we conclude that the existence of a polynomial time algorithm for the **Worst Risk Maximization Problem** implies $P = NP$. □

Corollary 1. *Generating an infinite sequence with optimal worst risk is intractable.*

Proof. A direct consequence of Theorem 1. □

Corollary 2. *Generating an infinite sequence with optimal worst relative risk is intractable.*

Proof. A direct consequence of Corollary 1 and Theorem 1 when $\delta > 0$. □

Corollary 3. *Generating the infinite sequence that leximin optimizes the vector of worst risks (or worst relative risks) of each beneficiary is intractable.*

Proof. We prove it by reduction from the **Worst Risk Maximization Problem**, since the sequence S that leximin-optimizes the vector $[\min_t Risk_1(S,t) \dots \min_t Risk_{n_b}(S,t)]$ also optimizes $\min_b \min_t Risk_b(S,t)$. The proof for relative risks is similar. □

4 Algorithms

The most intuitive solution to optimizing worst risk(s) is to try to keep all risks as large as possible at all times, i.e. greedily choose the action that leximin optimizes risks during the next step (choose the action with the best immediate effects). We will refer to this strategy as **GR** (Greedy leximin-optimizing next-step Risks). One weakness of this approach is that it assumes the world ends at the next step. In Example 1 it will always chose action 3, leading to arbitrarily bad risks for both beneficiaries if the game lasts long enough. Note that $F^\star_3 = 0$, meaning action 3 should not be played at all. One can easily extend the greedy heuristic to ignore the "unusable" actions, but that is not enough to prevent risks from getting arbitrarily bad. Consider Example 2, with $U^\star = [240, 240, 240]$ and unique $F^\star = [\frac{4}{15}, \frac{6}{15}, \frac{5}{15}]$. For this problem **GR** will chose action 1 repeatedly, leading to arbitrarily bad risks for the first and last beneficiaries. The reason is that action 2 and 3 hurt one of those beneficiaries more than action 1 hurts any of them, and **GR** lacks the look-ahead to see the benefits of using actions 2 and 3.

Example 2.

		Beneficiaries		
		b1	*b2*	*b3*
	1	*-20*	*30*	*-20*
Actions	*2*	*60*	*30*	*-40*
	3	*-56*	*-60*	*64*

The algorithms we propose in this paper are based on the following observation. All beneficiaries' risks are zero at time t if the number of times each action j was used up to time t are all proportional to the corresponding components of some $F^\star$ vector (i.e. $k_j(S_t) = F^\star_j \times t$, $\forall j$). Ideally all $k_j(S_t) = F^\star_j \times t$ at all times, but since the k_j functions only take integer values, that is not always possible. It is intuitive that the risks cannot get very bad at any time t' if the relative counts (k_j/t') for how much each action was used up to time t' are close enough to the corresponding values in $F^\star$. There are many ways one can construct sequences S to keep the $k_j(S_t)$ values close enough to the $F^\star_j \times t$ values; we will present two simple methods, and derive risk upper and lower bounds for both of them.

Our methods completely ignore the actions j which are not "used" in $F^\star$ (i.e. $F^\star_j = 0$). Let $n^\star_a$ be the number of actions "used" in $F^\star$; for simplicity, we rename the actions (reorder the dimensions of the simplex) such that all actions used in $F^\star$ come before the other ones:

$$\forall j \in \{1 \dots n^\star_a\} : F^\star_j > 0 \wedge \tag{7}$$

$$\forall j \in \{n^\star_a + 1 \dots n_a\} : F^\star_j = 0 \tag{8}$$

Method 1 Intuitively, this method choses an action that, up to that point, has been used the least relative to how often the action should have been used. Consequently, an action j can be chosen at time $t+1$ if it has the smallest ratio $\frac{k_j(D_t)}{F^\star_j \times t}$, where D is the sequence of decisions produced by this method (so D_t contains all its past decisions). Without affecting the decision process, one can eliminate t from the denominator. We formally describe this method with the following notation:

$$d_t \in \left\{ j \le n^\star_a \middle| \forall i \le n^\star_a : \frac{k_j(D_{t-1})}{F^\star_j} \le \frac{k_i(D_{t-1})}{F^\star_i} \right\} \tag{9}$$

where all functions k_j are extended such that $k_j(D_0) = 0$, $\forall j \in \{1 \dots n_b\}$. Note that multiple actions could tie for the minimum.

Method 2 While the previous approach helps less-often chosen actions catch up to the others, the next approach choses actions that — if used — will get the least ahead of the others. The sequence of decisions D' for this method is described formally as follows:

$$d'(t) \in \left\{ j \le n^\star_a \middle| \forall i \le n^\star_a : \frac{k_j(D'_{t-1}) + 1}{F^\star_j} \le \frac{k_i(D'_{t-1}) + 1}{F^\star_i} \right\} \tag{10}$$

We will derive the risk bounds for the first method, then reduce the second method to the first.

Lemma 1. *$\forall j \in \{1 \dots n^\star_a\}$ and $\forall t \in \mathbb{N}$: $\frac{k_{d_t}(D_{t-1})}{F^\star_{d_t}} \le \frac{k_j(D_{t-1})}{F^\star_j}$*

Proof. The result follows directly from Equation 9. □

Lemma 2. *$\forall i, j \in \{1 \dots n^\star_a\}$ and $\forall t \in \mathbb{N}$: $\frac{k_i(D_t)-1}{F^\star_i} \le \frac{k_j(D_t)}{F^\star_j}$*

Proof. If $k_i(D_t) = 0$, then the inequality holds since the left hand side is strictly negative and the right hand side is positive. If $k_i(D_t) > 0$, let t' be the last time action i was used ($t' = \operatorname{argmax}_{\tau=1 \dots t} d_\tau = i$).

All functions $k_l(D_t)$ are monotonically increasing with t because $\forall l \in \{1 \dots n_a^\star\}$ and $\forall t \in \mathbb{N}$: $k_l(D_t) = k_l(D_{t-1}) + 1$ if $d_t = l$ and $k_l(D_t) = k_l(D_{t-1})$ otherwise.

Consequently, the following must hold:

$$\frac{k_i(D_t)-1}{F_i^\star} = \frac{k_i(D_{t'})-1}{F_i^\star} = \frac{k_{d_{t'}}(D_{t'})-1}{F_{d_{t'}}^\star}$$
$$= \frac{k_{d_{t'}}(D_{t'-1})}{F_{d_{t'}}^\star} \leq \frac{k_j(D_{t'-1})}{F_j^\star} \leq \frac{k_j(D_t)}{F_j^\star} \quad (11)$$

□

Theorem 2. *Regardless of how the first method breaks the ties, the following double inequality is guaranteed to hold $\forall b \in \mathbb{B}$ and $\forall t \in \mathbb{N}$:*

$$\sum_{i=1}^{n_a^\star} \min(X_{i,b},0) \leq Risk_b(D,t) \leq \sum_{i=1}^{n_a^\star} \max(X_{i,b},0) \quad (12)$$

Proof. Since $\sum_{i=1}^{n_a^\star} F_i^\star \times X_{i,b} = 0$, we can rewrite $Risk_b(D,t) = \sum_{i=1}^{n_a^\star} k_i(t) \times X_{i,b}$ as $Risk_b(D,t) = \sum_{i=1}^{n_a^\star} k_i(D_t) \times X_{i,b} - \frac{k_{d_t}(D_t)}{F_{d_t}^\star} \times \left(\sum_{i=1}^{n_a^\star} F_i^\star \times X_{i,b}\right)$. Equivalently:

$$\begin{aligned} Risk_b(D,t) &= \sum_{i=1}^{n_a^\star} \left(k_i(D_t) - F_i^\star \times \frac{k_{d_t}(D_t)}{F_{d_t}^\star} \right) \times X_{i,b} \\ &= \sum_{\substack{i=1 \\ i \neq d_t}}^{n_a^\star} \left(k_i(D_t) - F_i^\star \times \frac{k_{d_t}(D_t)}{F_{d_t}^\star} \right) \times X_{i,b} \\ &= \sum_{\substack{i=1 \\ i \neq d_t}}^{n_a^\star} \left(k_i(D_{t-1}) - F_i^\star \times \frac{k_{d_t}(D_{t-1})}{F_{d_t}^\star} \right) \times X_{i,b} \end{aligned} \quad (13)$$

It follows as a direct consequence of Lemma 1 and Lemma 2 that:

$$0 \leq k_i(D_{t-1}) - F_i^\star \times \frac{k_{d_t}(D_{t-1})}{F_{d_t}^\star} \leq 1 \quad (14)$$

The theorem follows directly from the combination of Equation 13 with Equation 14. □

Theorem 3. *Regardless of how the second method breaks ties, the following double inequality is guaranteed to hold for $\forall b \in \mathbb{B}$ and $\forall t \in \mathbb{N}$:*

$$-\sum_{i=1}^{n_a^\star} \max(X_{i,b},0) \leq Risk_b(D',t) \leq -\sum_{i=1}^{n_a^\star} \min(X_{i,b},0) \quad (15)$$

Proof. We submit the following observation regarding the first method: since $\frac{1}{f_i} > \frac{0}{f_j}$ $\forall i,j \in \{1 \dots n_a^\star\}$, no action can be chosen a second time before every action has been chosen once. This implies that $\forall j$: $k_j(D_{n_a^\star}) = 1$, i.e. each action is chosen exactly once during the first $n_a^\star$ decisions.

We now claim that the second method makes the same sequence of decisions as the first method, but shifted to the right by $n_a^\star$ positions. That is, $d'_t = d_{t+n_a^\star}$, provided corresponding ties are broken the same way. This is because the "+1" in the decision function for the second method covers the difference of $n_a^\star$ steps between the two methods for each beneficiary.

The first methods' risk at time step $n_a^\star$ is $Risk_b(D,n_a^\star) = \sum_{i=1}^{n_a^\star} X_{i,b}$, which corresponds to a $Risk_b(D',0) = 0$. After that the two methods make the same decisions, so they change the risks the same way (because if they both chose action j, then they both add X_j to their risks). Therefore,

$$Risk_b(D,n+t) = Risk_b(D',t) + \sum_{i=1}^{n_a^\star} X_{i,b} \quad (16)$$

The theorem follows directly from applying Equation 16 to Theorem 2. □

5 Discussion

We proposed two methods for choosing actions such that the risks are always lower-bounded. While **GR** leximin-optimizes next-step risks, our methods *greedily* optimize actions' usage *frequencies* (relative counts) relative to some optimal configuration $F^\star$; we will call our methods **GF**.

We denote with LB_1 and LB_2 the lower bounds on risks guaranteed by the two **GF** methods respectively.

$$LB_1 = \min_b \sum_{i=1}^{n_a^\star} \min(X_{i,b},0) \quad (17)$$

$$LB_2 = \min_b \sum_{i=1}^{n_a^\star} -\max(X_{i,b},0) \quad (18)$$

There is an obvious way to unify the two: compute LB_1 and LB_2 ahead of time, then use the most promising method. Therefore our best worst risk guarantee is:

$$WR \geq \max(LB_1, LB_2) \quad (19)$$

In order to compute the approximation ratio, we need an upper bound on the optimal worst risk. Based on the fact that some action must be chosen first, it holds that $WR \leq \max_j \min_b X_{j,b}$.

$$\varepsilon \leq \frac{\max(LB_1, LB_2)}{\max_j \min_b X_{j,b}} \tag{20}$$

The computational complexity of our algorithms is $O(\lg n_a^\star)$ per time step. This is because one can use a heap to store actions' $\frac{k_j[+1]}{F_j^\star}$ scores, since our algorithms change a single action's score at each time step.

5.1 Eliminating Unnecessary Actions

If the vector $F^\star$ is not unique, the particular choice of $F^\star$ influences both the time complexity (through $n_a^\star$), and also WR, the worst risk. Each beneficiary's worst risk is lower-bounded by the sum of his negative X values (or the negative sum of its positive X values), so eliminating an action can only improve the worst risk.

Although the locus of $F^\star$ is convex, the goal of finding the $F^\star$ with the most zeros, or the $F^\star$ with the configuration of zeros to optimize $\min(LB_1, LB_2)$ require the use of *mixed integer programming*, and thus exponential computation time. In lieu of finding the subset of actions leading to the best WR, one can go with a greedy, polynomial-time approach.

Lemma 3. *There must always exist $F^\star$ such that $n_a^\star \leq n_b$.*

Proof. We associate with each action an n_b-dimensional point whose coordinates are equal to that action's rewards for each beneficiary. We note that leximin-optimality implies pareto-optimality, so $U^\star$ must be on the convex hull of the n_a points. The result follows directly from the boundary case of Carathéodory's theorem [11]. □

Based on this result, one can eliminate at least $\max(n_a - n_b, 0)$ actions in a preprocessing phase, thus reducing the per-step complexity to $O(\lg(\min(n_a, n_b)))$. We sketch a simple algorithm for this task based on a particular constructive proof for Carathéodory's theorem (e.g. [6]). One can use the Gauss elimination algorithm (e.g. [10]) to find a non-trivial solution $\alpha_2, \ldots, \alpha_{n_a^\star}$ to the following system of n_b equations $(X_{i,b} - X_{1,b}) \times \alpha_i = 0$. Let $\alpha_1 = -\sum_{i=2}^{n_a^\star} \alpha_i$. Note that $\sum_{i=1}^{n_a^\star} \alpha_i = 0$ and $\sum_{i=1}^{n_a^\star} \alpha_i \times X_i = 0$. Let $\theta = \min_{1 \leq i \leq n_a^\star} F_i^\star / \alpha_i | \alpha_i > 0$. We compute a new $F^\star$ vector by subtracting $\theta \times \alpha_i$ from each $F_i^\star$. The new vector has at least one extra zero at position k where $F_k^\star / \alpha_k = \theta$. The complexity of the Gauss elimination is $O(n_a n_b^2)$, and there are at most $n_a - 1$ iterations, so the entire pre-process of eliminating actions can be done in $O(n_a^2 n_b^2)$.

5.2 Breaking Ties with GR

The lower-bounds on worst risks (Equations 17 and 18) make no assumptions on how ties are broken, which means they assume the ties are broken in the worst possible way. Finding the best way to break ties is NP-hard (see the proof of Theorem 1), so a greedy approach will have to do. The most obvious solution would be to use **GR** to break ties for **GF** (for a time complexity of $O(n_b \times n_a^\star)$ per time step).

The potentially frequent occurrence of ties is not necessarily a weakness of the **GF** methods. One can use the opportunity to pursue other goals while being guaranteed that the risks are held in check. For instance, one can try to address the S_1 versus S_4 paradox for Example 1. In this case one should break ties to optimize beneficiaries' *cumulative risks* (weighted by the probability the game ends at the next time step).

6 Related Work

The game-theoretic work in [2, 8, 9] is concerned with finding Nash equilibria that result in alternating joint-actions (also referred to as turn-taking), but these results were tailored for specific classes of 2-by-2 games.

The starting point in our research on long-term fairness was the work in [18] on "periodic policies." Their reward model comes in the form of a normal-form game, but the players are actually cooperative learning agents (rather than self-interested). The process consists of the learners playing selfishly to discover a pure Nash equilibrium, while being interrupted periodically to compare accumulated rewards. The player gaining the most (in the current Nash equilibrium and overall) has its action put off-limits until the others catch up. Alternatively [17], after the learners discover all pure Nash equilibria, they create a periodic policy consisting of those joint actions with the fairest outcomes. In the authors' examples the players have only two actions: a highly lucrative one and a social one they fall back on while waiting for the other(s) to catch up. Both of these greedy algorithms could lead to utility profiles arbitrarily worse than the optimum.

The **Worst Risk Maximization Problem** is actually a variant of the **Compact Vector Summation Problem** [5, 14, 16]. This problem states that given a finite set of vectors $X_1, \ldots, X_n \in \mathbb{R}^m$ such that $\sum_{i=1}^n X_i = 0$, one must find a permutation π of $\{1, 2, \ldots, n\}$ that minimizes $\max_{1 \leq k \leq n} \| \sum_{i=1}^k X_{\pi_i} \|$. In the earliest such work $\| \cdot \|$ was l^2, the Euclidian norm (i.e. $\|y\|_2 = \sqrt{\sum_{i=1}^m y_i^2}$), so the problem consisted of ordering the vectors such that the path resulting from adding vectors one by one stays inside a minimum-radius m-dimensional circle centered at the origin. Later research has focused on results general enough to accommodate arbitrary norms (intuitively a norm is a function associating a "size" value to every vector).

The best performance guarantee we are aware of was proven in [1]: there exists a permutation such that $\max_{1\leq k\leq n} \|\sum_{i=1}^{k} X_{\pi_i}\| \leq M(m-1+\frac{1}{m})$, where $M = max_{1\leq k\leq n}\|X_i\|$ is the size of the largest vector (with respect to that norm). Later on the result was extended to a relaxed version of norms called "asymmetric norms" (we define these concepts shortly) and an $O(n^2m^2)$ algorithm was proposed [14, 15].

We submit that this algorithm is the most relevant algorithm for a comparison with our algorithms: it has the best guarantee and time complexity and it accommodates "asymmetric norms." The last item is relevant because comparing algorithms is more meaningful if they actually try to optimize the same function, and the function we try to optimize can be rearranged as an asymmetric norm but not a norm. Maximizing the worst risk is equivalent to minimizing is the largest absolute value of any negative coordinate of any partial sum of X vectors. This function is an asymmetric norm, but not a norm, since it satisfies the triangle inequality ($\|y+z\| \leq \|y\|+\|z\|$) and positive definiteness ($\|y\| = 0 \Rightarrow y = 0$), but it only satisfies the scalability condition ($\|ky\| = |k| \times \|y\|$) for *positive scaling* factors [3, 13].

Guarantee Comparison We submit that $|LB_1| \leq M(n_a^\star - 1)$. This results from Equation 17 by replacing every negative $X_{i,b}$ with $-M$, the smallest coordinate of any X vector, and noticing that for any given beneficiary b at most $n_a^\star - 1$ of his X values can be negative.[3] Therefore the absolute value of our worst risk, $|WR| \leq |LB_1| \leq M(n_a^\star - 1) = M(\min(n_a, n_b) - 1) < M(n_b - 1 + \frac{1}{n_b})$, which is the guarantee of Sevast'janov's algorithm.

Complexity Comparison Sevast'janov's algorithm picks a vector $n-m$ times, and each such operation has a complexity of $O(km^2)$, where k is the number of alternatives ($k = n, \ldots, n-m$). An iteration in Sevast'janov's algorithm has the same complexity as an iteration in our preprocessing phase (they're both based on Gauss elimination). We also submit that the number of iterations in Sevast'janov's algorithm is at least the number of iteration in our preprocessing phase (each action has at least multiplicity 1). Therefore the time complexity of our preprocessing phase cannot be larger than that of Sevastianov's algorithm. More importantly, even if that algorithm were extended to benefit from our preprocessing phase and to explicitly deal with multiplicities (i.e. $k = n_a^\star$), its complexity would still be $O(n_a^\star n_b^2)$ per time step which is an order of magnitude higher than the complexity of our algorithms when breaking ties with **GR**.

In summary, by eliminating unnecessary actions and only keeping track of multiplicities, we are able to offer worst case guarantees that are never worse than (and sometimes much better than) Sevast'janov's.

7 Future Work

It turns out that if two beneficiaries get the exact same rewards out of every action, eliminating one of them has no effect on the algorithms proposed here. The $U^\star$ vector stays the same (the algorithm computing $U^\star$ acts as constructive proof), and so does the lower bound on risks (Equation 17 and Equation 18). Therefore one can see n_b as the number of user *classes* rather than the number of *users* in one's system.

This observation might prove very helpful with respect to both quality of guarantees and computational complexity if there are lots of beneficiaries, which belong to relatively few classes. We plan to investigate the implications of clustering together beneficiaries with similar, but not identical reward profiles.

We also intend to modify the action elimination algorithm. In a first stage we wish to find a computationally inexpensive way to bias it towards eliminating specific actions (e.g. those the negatively affect LB_1 or LB_2 the most). In the second stage we wish to find an approximation scheme for finding feasible action subsets whose $\min(LB_1, LB_2)$ stays within a certain factor from the optimum.

8 Conclusions

In this paper we studied the problem of achieving certain long-term fairness guarantees in a simple repeated-game setup: (1) all beneficiaries are entitled to their socially-optimal utilities and (2) no matter when the game ends all beneficiaries are guaranteed to have received close to what they were entitled to that point. We proved that finding an optimal solution with respect to the second guarantee is NP-hard and proposed two efficient approximation algorithms.

9 Acknowledgements

We thank Octav Olteanu for our helpful discussions, Joey Harrison for his help with putting together this article and Zoran Duric, Alexei Samsonovich, and Alexander

[3] From Equation 6 it follows that for any beneficiary b, if there exists $i \in \{1 \ldots n_a^\star\}$ such that $X_{i,b} < 0$ than there must exist $j \in \{1 \ldots n_a^\star\}$ such that $X_{j,b} > 0$.

Brodsky for helping us with the related work literature in Russian.

References

[1] Wojciech Banaszczyk. The Steinitz constant of the plane. *Journal für die reine und angewandte Mathematik*, 373:218–220, 1987.

[2] V. Bhaskar. Egalitarianism and efficiency in repeated symmetric games. *Games and Economic Behavior*, 32(2):247–262, August 2000.

[3] P. A. Borodin. The Banach-Mazur theorem for spaces with asymmetric norm. *Mathematical Notes*, 69:298–305, 2001.

[4] Didier Dubois, Philippe Fortemps, Marc Pirlot, and Henri Prade. Leximin optimality and fuzzy set-theoretic operations. *European Journal of Operational Research*, 130(1):20–28, 2001.

[5] Steven R. Finch. *Mathematical Constants*. Cambridge University Press, 2003.

[6] Monique Florenzano and Cuong Le Van. *Finite Dimensional Convexity and Optimization*. Springer, 2001.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, January 1979.

[8] Sau-Him Paul Lau and Vai-Lam Mui. Achieving intertemporal efficiency and symmetry through intratemporal asymmetry: (eventual) turn taking in a class of repeated mixed-interest games. Technical Report 1092, Hong Kong Institute of Economics and Business Strategy, November 2003.

[9] Sau-Him Paul Lau and Vai-Lam Mui. Using turn taking to mitigate conflict and coordination problems in the battle of the sexes game. Technical Report 1129, Hong Kong Institute of Economics and Business Strategy, April 2005.

[10] Seymour Lipschutz and Marc Lipson. *Schaum's Outline of Theory and Problems of Linear Algebra*. McGraw-Hill, Inc., New York, NY, USA, third edition, 2001.

[11] Robert R. Phelps. *Lectures on Choquets Theorem*. Springer, 2001.

[12] Jos A. M. Potters and Stef H. Tijs. The nucleolus of a matrix game and other nucleoli. *Mathematics of Operations Research*, 17(1):164–174, 1992.

[13] H. H. Schaefer. *Topological Vector Spaces*. Number 3 in Graduate Texts in Mathematics. Springer-Verlag, second edition, 1999.

[14] Sergey Sevast'janov. On the compact summation of vectors. *Diskretnaya Matematika*, 3(3):66–72, 1991. in Russian.

[15] Sergey Sevast'janov. Vector summation in Banach space and polynomial algorithms for flow shops and open shops. *Math. Oper. Res.*, 20(1):90–103, 1995.

[16] Sergey Sevast'janov and Wojciech Banaszczyk. To the Steinitz lemma in coordinate form. *Discrete Mathematics*, 169(1):145–152, 1997.

[17] Katja Verbeeck, Ann Nowé, Johan Parent, and Karl Tuyls. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Journal of Autonomous Agents and Multi-Agent Systems*, 14(3):239–269, 2006.

[18] Katja Verbeeck, Johan Parent, and Ann Nowé. Homo egualis reinforcement learning agents for load balancing. In *Innovative Concepts for Agent-Based Systems: 1^{st} International Workshop on Radical Agent Concepts*, volume 2564 of *Lecture Notes in Computer Science*, pages 81–91, January 2003.