STAIRS 2012
K. Kersting and M. Toussaint (Eds.)
© 2012 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-096-3-204

Complexity and Approximability of Egalitarian and Nash Product Social Welfare Optimization in Multiagent Resource Allocation[1]

Nhan-Tam NGUYEN [a,2], Trung Thanh NGUYEN [a], Magnus ROOS [a], and Jörg ROTHE [a]

[a] *Institut für Informatik*
Heinrich-Heine-Universität Düsseldorf
40225 Düsseldorf, Germany

Abstract. Multiagent resource allocation deals with distributing (bundles) of resources to agents that specify utility functions over bundles. A natural and important problem in this field is social welfare optimization. We assume resources to be indivisible and nonshareable and that utility functions are represented by the k-additive form or as straight-line programs. We prove NP-completeness for egalitarian and Nash product social welfare optimization for straight-line program representation of utility functions. In addition, we show that social welfare optimization by the Nash product in the 1-additive form is hard to approximate, yet we also give fully polynomial-time approximation schemes for egalitarian and Nash product social welfare optimization in the 1-additive form with a fixed number of agents.

Keywords. Multiagent resource allocation, social welfare optimization, computational complexity, economically motivated agents, auctions, approximability

1. Introduction

Resource allocation is at the core of multiagent research. In the resource allocation framework we deal with the following scenario. There are autonomous (software) agents with individual goals and utilities. A central entity distributes indivisible and nonshareable resources to the agents, each agent assigning utility to bundles of resources. As designers of such systems we may have different goals, e.g., ensuring a high average well-being of every agent or maintaining fairness among the agents. Depending on our goal, we differently aggregate the utilities that each agent realizes upon receiving some bundle. On the one hand, ensuring a high well-being overall usually means resorting to

[1] Supported in part by DFG under grant RO-1202/14-1, ARC under grant DP110101792, a DAAD PPP grant in the PROCOPE program, and a fellowship from the Vietnamese government.

[2] Corresponding Author: Nhan-Tam Nguyen, Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, Universitätsstr. 1, 40225 Düsseldorf, Germany; E-mail: Nhan-Tam.Nguyen@uni-duesseldorf.de.

utilitarian social welfare, i.e., the sum of the agents' utilities. On the other hand, maximizing the minimum utility that the agents realize, that is, maximizing *egalitarian social welfare*, might increase fairness to some extent. A simple approach to bridge these two conflicting goals is the aggregation of utilities by taking their product. Maximizing this *Nash product* means achieving a "balanced" utility vector.

Because we have software agents, we have to pay attention to the encoding of their utility functions. Naive approaches such as the *bundle form* (see, e.g., [3]), where we list all bundles with their attached utility (dropping bundles with zero utility) easily lead to an exponential blowup. The *k-additive form* succinctly captures the interrelated nature of resources, which explains why agents express their utilities over *bundles* of resources.[3] There are other succinct representation forms as well, e.g., by interpreting utility functions as computational processes. The idea underlying the *straight-line program* (SLP) representation is to express a given utility function by a shortest program that can efficiently compute it. This may result in compact representation of utility functions as well.

We study the computational complexity of social welfare optimization problems when utility functions are represented by straight-line programs, an issue left open by Roos and Rothe [14]. We show that egalitarian and Nash product social welfare optimization are NP-complete in this case. In addition, we look at the 1-additive representation of utility functions and present an inapproximability result for Nash product social welfare optimization. However, we also show that egalitarian social welfare optimization and Nash product social welfare optimization admit fully polynomial-time approximation schemes when the number of agents is fixed.

2. Formal Model

2.1. Multiagent Resource Allocation Settings

Following Chevaleyre et al. [3], let $A = \{a_1, \dots, a_n\}$ denote the set of n *agents* and $R = \{r_1, \dots, r_m\}$ the set of m indivisible and nonshareable *resources*. Every agent a_i specifies a *utility function* u_i, which is a mapping from the power set of resources, 2^R, to some numerical set such as $\mathbb{Q}$, the set of rational numbers, or $\mathbb{Q}^+$, the set of nonnegative rational numbers. Let U be the set of all utility functions for the agents in A. A triple (A, R, U) is called a *multiagent resource allocation setting* (for short, *MARA setting*). An *allocation* of resources to the agents is modeled by a function $X : A \to 2^R$ such that for all distinct $a_i, a_j \in A$ we have $X(a_i) \cap X(a_j) = \emptyset$ and $\bigcup_{a_i \in A} X(a_i) = R$. That is, a resource is given exclusively to a single agent and all resources are given to some agents. The set of all allocations for a MARA setting (A, R, U) is denoted by $\Pi_{A,R}$ and has cardinality n^m.

2.2. Representing Utility Functions

We consider two possible ways to represent utility functions, the k-additive form and the straight-line program representation.

[3]Agents may find a bundle of several items worth more, or less, than the sum of the utilities of each item individually. For example, think of complementary items such as a left and a right shoe (which may be almost worthless when bought alone but very valuable together), or of discounts for bulk purchases.

For some fixed $k \in \mathbb{N}$, a utility function u is in *k-additive form* if there are unique coefficients α_T for every $T \subseteq R$ with $\|T\| \leq k$ such that for every bundle of resources R' we have

$$u(R') = \sum_{T \subseteq R', \|T\| \leq k} \alpha_T.$$

On the one hand, if k is large enough, we can represent every utility function. On the other hand, for small k, utility functions can be encoded succinctly. The k-additive form has been introduced in multiagent resource allocation by Chevaleyre et al. [4,5] and, independently, in combinatorial auctions by Conitzer et al. [6]. We only deal with the 1-additive form, that is, only with additive utility functions.

The other representation form that we consider is the straight-line program representation. Informally, a *straight-line program* is a topologically sorted list of gates of a boolean circuit C that takes as input an m-dimensional binary vector and outputs s bits. Interpreting the input vector as a bundle of resources R' and the output as the binary representation of $u(R')$, we can say that C (or a corresponding straight-line program) represents utility function u.

Formally (see [7]), an *(m,s)-combinational logic network* is a directed graph with m input nodes $(\beta_1, \ldots, \beta_m)$ of in-degree 0, s output nodes $(\gamma_{s-1}, \ldots, \gamma_0)$ of out-degree 0, and gate nodes of in-degree at most 2 and out-degree at least 1. A gate node represents one of the common boolean operations $(\wedge, \vee, \neg)$. An input to the nodes $(\beta_1, \ldots, \beta_m)$ can be interpreted as a vector of length m and vice versa. Hence, every input vector β induces[4] an output vector $C(\beta)$, where we denote by $C(\beta)_i$ the i-th least significant bit of $C(\beta)$. Let $R = \{r_1, \ldots, r_m\}$, let $u : R \to \mathbb{N}$ be a utility function and C an (m,s) combinational logic network. Denote by β_S the characteristic vector that has for every $j \in \{1, \ldots, m\}$ the j-th coordinate equal 1 if and only if $r_j \in S$ for some $S \subseteq R$. Utility function u is *realized* by C if for every $S \subseteq R$ with binary vector β_S the following holds:

$$u(S) = \sum_{i=0}^{s-1} 2^i \cdot C(\beta_S)_i.$$

The advantages of straight-line programs are mainly the efficiency of evaluation (linear time in the number of nodes) and its conciseness, which is supported by the following result by Pippenger and Fischer [12] and Schnorr [16].

Fact 1 *Let $f : \{0,1\}^m \to \{0,1\}^s$. If there exists a deterministic Turing machine that computes f in time T, then there exists a straight-line program of $\mathcal{O}(T \log T)$ lines that computes f as well.*

In multiagent resource allocation, utility representations by straight-line programs were introduced by Dunne et al. [7].

[4]Every bit at a gate node is induced as usual: If a is a gate node with a 2-ary boolean operation σ, then the bit induced at a is $b_1 \sigma b_2$, provided that (b_1,a) and (b_2,a) are edges of the graph, σ is a binary operation, and by b_1 and b_2 we mean the induced bits at nodes b_1 and b_2. For the boolean operation $\neg$, the definition is analogous.

2.3. Notions of Social Welfare

Depending on the objective of the system designer, different notions of social welfare are appropriate. We will study the following notions of social welfare.

Definition 1 *For a MARA setting (A,R,U) and an allocation $X \in \Pi_{A,R}$, define*

1. *the* utilitarian social welfare of *X as $sw_u(X) = \sum_{a_i \in A} u_i(X)$;*
2. *the* egalitarian social welfare of *X as $sw_e(X) = \min_{a_i \in A}\{u_i(X)\}$;*
3. *the* Nash product of *X as $sw_N(X) = \prod_{a_i \in A} u_i(X)$.*

2.4. Social Welfare Optimization Problems

We consider two problem types of social welfare optimization, decision problems and maximization problems. For $\mathbb{F} \in \{\mathbb{Q}^+, \mathbb{Q}\}$ and form $\in \{k\text{-add} \mid k \geq 1\} \cup \{\text{SLP}\}$, where k-add abbreviates "k-additive" and SLP "straight-line program," define the decision problem:

$\mathbb{F}$-Egalitarian Social Welfare Optimization$_{\text{form}}$	
Given:	A MARA setting $M = (A,R,U)$, where form indicates how every $u_i : 2^R \to \mathbb{F}$ in U is represented, and a threshold $t \in \mathbb{F}$.
Question:	Is there an allocation $X \in \Pi_{A,R}$ such that $sw_e(X) \geq t$?

We abbreviate this problem by $\mathbb{F}$-ESWO$_{\text{form}}$. The corresponding problems for utilitarian and Nash product social welfare can be defined analogously and are abbreviated by $\mathbb{F}$-USWO$_{\text{form}}$ and $\mathbb{F}$-NPSWO$_{\text{form}}$, respectively.

The maximization problem for utilitarian social welfare is formally defined as follows:

$\mathbb{F}$-Maximum Utilitarian Social Welfare$_{\text{form}}$	
Input:	A MARA setting $M = (A,R,U)$, where form indicates how every $u_i : 2^R \to \mathbb{F}$ in U is represented.
Output:	$\max\{sw_u(X) \mid X \in \Pi_{A,R}\}$.

As a shorthand, write $\mathbb{F}$-Max-USW$_{\text{form}}$. Based on sw_e and sw_N, $\mathbb{F}$-Maximum Egalitarian Social Welfare$_{\text{form}}$ (or $\mathbb{F}$-Max-ESW$_{\text{form}}$) and $\mathbb{F}$-Maximum Nash Product Social Welfare$_{\text{form}}$ (or $\mathbb{F}$-Max-NPSW$_{\text{form}}$) are defined accordingly.

2.5. Background on Approximation Theory

We define the relevant notions of approximation theory, i.e, approximation algorithms, polynomial-time approximation schemes, and reducibilities to prove hardness of approximation.

Definition 2 (α-approximation algorithm) *Let* Π *be an maximization problem and* $\alpha : \mathbb{N} \to [0,1]$ *be a polynomial-time computable function. An* α-approximation algorithm A *for* Π *is a polynomial-time algorithm such that for each instance x of* Π*,*

$$val(A(x)) \geq \alpha(|x|) \cdot OPT(x),$$

where $val(A(x))$ *denotes the value of a solution produced by A on input x and where* $OPT(x)$ *denotes the value of an optimal solution for x.*

The *approximation factor* α might be a constant function such as $1-\varepsilon$ for some ε, $0 < \varepsilon < 1$, or a function of the input size.

Definition 3 (FPTAS) *A maximization problem* Π *has a* fully polynomial-time approximation scheme *(*FPTAS*) if for each* ε*,* $0 < \varepsilon < 1$*, there exists a* $(1-\varepsilon)$*-approximation algorithm* A_ε *for* Π*, where the running time is polynomial in* $1/\varepsilon$ *as well.*

One approach to prove inapproximability for a maximization problem is to find an α-gap-introducing reduction from an NP-complete problem. This is also known as *producing hard gaps*.

Definition 4 (α-gap-introducing reduction) *Let* $A \subseteq \Sigma^*$ *be an* NP*-complete problem,* Π *be a maximization problem, and let* $\alpha : \mathbb{N} \to [0,1]$ *be a polynomial-time computable function of the input size. An* α-gap-introducing reduction from A to Π *is given by two polynomial-time computable functions f and g such that for each* $x \in \Sigma^*$*,*

1. $g(x)$ *is an instance of* Π*,*
2. *if* $x \in A$ *then* $OPT(g(x)) \geq f(x)$*, and*
3. *if* $x \notin A$ *then* $OPT(g(x)) < \alpha(|x|) \cdot f(x)$*.*

Note that an α-approximation algorithm B for a maximization problem Π that has an α-gap-introducing reduction from an NP-complete problem A implies $x \in A$ if and only if the value of the solution $B(g(x))$ is at least $\alpha(|x|) \cdot f(x)$. Hence, there can be no α-approximation algorithm for Π, unless P = NP.

Definition 5 (L-reduction) *Let* Π_1 *and* Π_2 *be some maximization problems. An* L-reduction from Π_1 to Π_2 *is given by two polynomial-time computable functions f and g and two parameters* α *and* β *such that for each instance x of* Π_1*,*

1. $y = f(x)$ *is an instance of* Π_2*,*
2. $OPT(y) \leq \alpha \cdot OPT(x)$*, and*
3. *for each solution* s_2 *for y of value* v_2*,* $s_1 = g(s_2)$ *is a solution for x of value* v_1 *such that*

$$OPT(x) - v_1 \leq \beta \cdot (OPT(y) - v_2).$$

Having an L-reduction from maximization problem Π_1 to Π_2 with parameters α,β and an $(1-\varepsilon)$-approximation algorithm for Π_2 implies a $(1-\alpha\beta\varepsilon)$-approximation algorithm for Π_1 by invoking f on the instance x of Π_1 to get an instance y of Π_2, then running the approximation algorithm for Π_2 on y and, at last, translating the solution back via g. Note that if Π_1 does not admit a $(1-\varepsilon)$-approximation algorithm and reduces to

Π_2 with parameters $\alpha = \beta = 1$ then Π_2 cannot have a $(1-\varepsilon)$-approximation algorithm either.

For more background on approximation theory, see, e.g., the textbook by Vazirani [17] and the survey by Arora and Lund [1].

3. Related Work

Chevaleyre et al. [4] (see also [5]) were the first to show that utilitarian social welfare optimization in MARA is NP-complete for the "bundle form" representation of a utility function (a simple enumerative listing of all subsets that have nonzero utility under the utility function being represented) and the k-additive form. For the straight-line program representation, Dunne et al. [7] proved the same problem to be NP-complete as well.

Roos and Rothe [14] showed NP-completeness of egalitarian and Nash product social welfare optimization for both the bundle form and the k-additive form. In addition, they studied the complexity of the exact variants of social welfare optimization problems. Lipton et al. [10] provided a reduction to prove NP-hardness of finding a minimum-envy allocation (i.e., an allocation X that minimizes the envy $\max_{i,j}\{0, u_i(X(a_j)) - u_i(X(a_i))\}$). This reduction proves NP-hardness of the decision problem associated with egalitarian social welfare optimization as well. Independently of Roos and Rothe [14], Ramezani and Endriss [13] proved the same NP-completeness result of Nash product social welfare optimization for the bundle form.

For results on the approximability of social welfare optimization, see the survey by Nguyen, Roos, and Rothe [11].

4. Results

Our first result is the NP-completeness of egalitarian and Nash product social welfare optimization when representing utility functions as straight-line programs. The intuition is to encode a formula into the utility function of an agent.

Theorem 1 $\mathbb{Q}$-ESWO$_{\text{SLP}}$ *and* $\mathbb{Q}^+$-NPSWO$_{\text{SLP}}$ *are* NP-*complete.*

Proof. Membership in NP is easy to see. To show NP-hardness, we reduce from the NP-complete problem MAX3SAT, which is formally defined as follows:

MAX3SAT	
Given:	A boolean formula φ in 3-CNF (i.e., in conjunctive normal form with three literals per clause) and $k \geq 2$.
Question:	Is there an assignment to the variables of φ such that at least k clauses are satisfied?

Let $\varphi = \bigwedge_{i=1}^{m}(z_i^1 \vee z_i^2 \vee z_i^3)$ be a given boolean formula in 3-CNF, where z_i^j, $1 \leq i \leq m$ and $j \in \{1,2,3\}$, is a literal of some variable $v \in V = \{v_1, \ldots, v_n\}$. Define $A = \{a_1, a_2\}$

and $R = \{r_1, \dots, r_n, r_{n+1}, \dots, r_{2n}\}$. We say a bundle $S \subseteq R$ or its corresponding vector $\alpha_S = (x_1, \dots, x_n, x_{n+1}, \dots, x_{2n})$ is *valid* if

$$\bigwedge_{i=1}^{n} XOR(x_i, x_{n+i}) = \bigwedge_{i=1}^{n} (\neg x_i \wedge x_{n+1}) \vee (x_i \wedge \neg x_{n+i}) = 1,$$

i.e., *XOR* denotes the boolean exclusive-or operation. Define a_1's utility function as

$$u_1(S) = \begin{cases} \text{number of satisfied clauses in } \varphi \text{ by } S & \text{if } S \text{ is valid} \\ 0 & \text{otherwise} \end{cases}$$

and a_2's utility function as

$$u_2 \equiv \begin{cases} m & \text{if we reduce to the egalitarian social welfare} \\ 1 & \text{if we reduce to social welfare by the Nash product.} \end{cases}$$

Write

$$u_1(\alpha_S) = \left(\bigwedge_{i=1}^{n} XOR(x_i, x_{n+i}) \right) \cdot \sum_{i=1}^{m} (z_i^1 \vee z_i^2 \vee z_i^3),$$

where we replace[5] z_i^j by the corresponding value of x_k, $k \in \{1, \dots, n\}$, if z_i^j is a positive literal of x_k; otherwise (that is, if z_i^j is a negated variable) we replace it by the value of x_{n+k}, $k \in \{1, \dots, n\}$. By Proposition 1, we know there is an SLP of polynomial size that represents u_1.

Now consider a 3-CNF formula φ whose maximum number of satisfied clauses is k for some assignment $A : X \to \{0,1\}$. Assignment A induces an assignment vector $\alpha_S = (A(v_1), \dots, A(v_n), 1 - A(v_1), \dots, 1 - A(v_n))$. By definition, α_S is valid and a_1's utility is exactly k. The remaining resources go to a_2. Because a_2's utility can be ignored, the social welfare is a_1's utility, that is, the maximum number of satisfied clauses in φ.

For the other direction, note that we reduced from a legal 3-CNF formula. So there is an assignment that satisfies at least one clause. Hence, a_1 realizes a utility of at least one. Now let $k \geq 1$ be the maximum social welfare of this instance. By definition, $u_1(S) = k$ for some valid $\alpha_S = (x_1, \dots, x_n, x_{n+1}, \dots, x_{2n})$. Truncating α_S by dropping the last n coordinates yields an assignment that satisfies k clauses. ❑

Notice that the reduction in the proof of Theorem 1 is an L-reduction with parameters $\alpha = \beta = 1$. There is a one-to-one correspondence between assignments of variables and assignments of resources to the first agent where the maximum number of satisfied clauses equals the social welfare after the reduction. By setting the utility function of the second agent to the constant zero-function, we have a reduction with the same properties for the utilitarian case. Using the inapproximability result of Max3SAT by Håstad [8] we conclude:

[5]Because we have a boolean circuit, we actually insert an edge (x_k, o_p^q), where o_p^q, $p \in \{1, \dots, m\}$, $q \in \{1,2\}$, denotes the $\vee$-gate that is responsible for z_i^j in clause p.

Corollary 1 $\mathbb{Q}$-MAX-USW$_{\text{SLP}}$, $\mathbb{Q}$-MAX-ESW$_{\text{SLP}}$, $\mathbb{Q}^+$-MAX-NPSW$_{\text{SLP}}$ *are* NP-*hard to approximate within a factor of* $7/8+\varepsilon$ *for every* $\varepsilon > 0$.

The next result deals with the hardness of approximability for Nash product social welfare optimization for 1-additive utility functions. We prove this result by a reduction from the well-known NP-complete problem EXACT COVER BY THREE SETS, which is defined as follows:

EXACT COVER BY THREE SETS (X3C)	
Given:	A finite set B with $\|B\| = 3n$ and a collection $C = \{S_1,\dots,S_m\}$ of 3-element subsets of B.
Question:	Does there exist a subcollection $C' \subseteq C$ such that every element of B occurs in exactly one of the sets in C'?

Theorem 2 *Assuming* P $\neq$ NP, MAX-NPSW$_{1\text{-}additive}$ *cannot be approximated within a factor of* $2/3+\varepsilon$ *for any* $\varepsilon > 0$.

Proof. Let (B,C) with $\|B\| = 3n$ and $C = \{S_1,\dots,S_m\}$ be an instance of X3C. Without loss of generality, assume that $m \geq n$. Construct an instance $M = (A,R,U)$ of $\mathbb{Q}^+$-MAX-NPSW$_{\text{1-additive}}$ as follows. Let A be a set of m agents, where agent a_i corresponds to S_i, and let $R = B \cup D$ be a set of $2n+m$ resources. That is, there are $3n$ "real" resources that correspond to the $3n$ elements of B, and there are $m-n$ "dummy" resources in D. Define the agents' utilities as follows. For each $a_i \in A$ and each $r_j \in R$, let

$$u_i(r_j) \;=\; \begin{cases} 1/3 & \text{if } r_j \in S_i \\ 1 & \text{if } r_j \in D \\ 0 & \text{otherwise.} \end{cases}$$

Also, define $u_i(\emptyset) = 0$ for all i, $1 \leq i \leq m$.

Suppose that (B,C) is a yes-instance of X3C. Then there exist a set $I \subseteq \{1,\dots,m\}$, $\|I\| = n$ such that $S_i \cap S_j = \emptyset$ for all $i,j \in I$, $i \neq j$, and $\bigcup_{i\in I} S_i = B$. Hence, we assign the bundle S_i to agent a_i for each $i \in I$, and the dummy resource to the $m-n$ remaining agents. This allocation maximizes the Nash product social welfare, which now is at least 1. Furthermore, the sum of all agents' utilities is at most m. Hence, the product of the agents' individual utilities is maximal if and only if all agents have the same utility, which exactly equals 1.

Conversely, if (B,C) is a no-instance of X3C, we show that the maximum Nash product social welfare is at most $2/3$. Obviously, the sum of all agents' utilities is at most $m-1/3$ in this case. The Nash product social welfare reaches the maximal value iff the utilities of the agents are as balanced as possible. The best allocation that satisfies this property is the following. Dummy resources are distributed to $m-n$ agents, $n-1$ agents get the $n-1$ disjoint bundles from $(S_1,\dots,S_m)$, and the last agent is assigned the remaining bundle which has utility of at most $2/3$. This implies that $\max_N(M) \leq 2/3$. Therefore, an approximation algorithm with a factor better than $2/3$ will distinguish the "yes" and "no" instances of X3C. ❑

Theorem 2 shows that MAX-NPSW$_{\text{1-additive}}$ cannot have a PTAS unless P $=$ NP. This result also holds for MAX-ESW$_{\text{1-additive}}$ due to Bezáková and Dani [2]. However, we now show that there is an FPTAS for this problem whenever the number of agents is fixed, using a technique that was also used to give an FPTAS for a variety of scheduling problems (see [15] and [9]). From now on, we assume that for any agent a_i, the utility function u_i is nonnegative and $u_i(\emptyset) = 0$.

Theorem 3 *Both* MAX-NPSW$_{\textit{1-additive}}$ *and* MAX-ESW$_{\textit{1-additive}}$ *admit an FPTAS for any fixed number of agents.*

Proof. Let $M = (A,R,U)$ be a MARA setting with 1-additive utilities and a fixed number $n = \|A\|$ of agents. As a shorthand, we denote by s_{ij} the utility of resource r_j for agent a_i for i, $1 \leq i \leq n$, and j, $1 \leq j \leq m$.

The proof of this theorem will be divided into two parts. In the first part, we construct a pseudo-polynomial time algorithm for MAX-NPSW$_{\text{1-additive}}$ that runs in time $\mathscr{O}(mB^n)$ where $B = \max_{1 \leq i \leq n} \sum_{j=1}^{m} s_{ij}$. We then prove in the second part that this algorithm yields a fully polynomial-time approximation scheme for our problem.

Let $T = (e_1, \dots, e_n)$ be a canonical basis of the vector space $\mathbb{R}^n$, where e_i denotes the vector with a 1 in the i-th coordinate and 0's elsewhere. Now, consider Algorithm 1.

Algorithm 1 Pseudo-polynomial-time algorithm

```
1: $V_0 := \{e_i \mid 1 \leq i \leq n\}$
2: for $j := 1$ to $m$ do
3:     $V_j := \emptyset$
4:     for each $v \in V_{j-1}$ do
5:         $V_j := V_j \cup \{v + s_{ij} \cdot e_i \mid i = 1, \dots, n\}$
6:     end for
7: end for
8: return Vector $v \in V_m$ that has the maximal product of its coordinates.
```

Clearly, $\|V_m\| > \|V_j\|$ for all j, $1 \leq j \leq m-1$. Furthermore, we have $\|V_m\| \leq B^n$ since the coordinates of all vectors of V_m are integers which do not exceed B. Hence, the running time of above algorithm is in $\mathscr{O}(m \sum_{k=1}^{m} \|V_k\|) = \mathscr{O}(mB^n)$.

We make a small modification to the above pseudo-polynomial-time algorithm. In more detail, we will remove some unnecessary vectors from V_j, for all j, $1 \leq j \leq m$. This implies that the algorithm may perhaps not return the exact optimal solution but it will give a good approximation to the solution. Indeed, let ε be any fixed positive number such that $0 < \varepsilon < 1$, and consider Algorithm 2.

Let $V \subseteq \mathbb{N}^n$, $K = \lceil \log_\alpha B \rceil$ and $L_i = [\alpha^{i-1}, \alpha^i]$ for i, $1 \leq i \leq K$. We define a relation $\sim$ on the set V as follows. For any two vectors $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ in V, $x \sim y$ if for every i, $1 \leq i \leq n$, $x_i = y_i = 0$ or $x_i, y_i \in L_j$ for some $j \in \{1, \dots, K\}$. Obviously, this relation is reflexive, symmetric and transitive and thus, it is an equivalence relation on V. Moreover, under this relation, V can be partitioned into equivalence classes, i.e., any two vectors from the same class are equivalent with respect to $\sim$. We claim that if $x \sim y$ then $x_i \geq (1/\alpha) y_i$ for all i, $1 \leq i \leq n$. Indeed, the statement is obviously true if $x_i = y_i = 0$. So, consider all other $x_i, y_i \in L_j$, that is, $\alpha^{j-1} \leq x_i, y_i \leq \alpha^j$. In this case we have $x_i/y_i \geq \alpha^{j-1}/\alpha^j = 1/\alpha$. We now prove by induction on j that for every vector

Algorithm 2 FPTAS for MAX-NPSW$_{\texttt{additive}}$

```
1: $\alpha := 1 + \varepsilon/2nm$
2: $K := \lceil \log_\alpha B \rceil$
3: $L_i = [\alpha^{i-1}, \alpha^i]$ for $i = 1, \dots, K$
4: $V_0^* := \{e_i \mid 1 \le i \le n\}$
5: for $j := 1$ to $m$ do
6:     $V_j^* := \emptyset$
7:     for each $v^* \in V_{j-1}^*$ do
8:         $V_j^* := V_j^* \cup \{v^* + s_{ij} \cdot e_i \mid i = 1, \dots, n\}$
9:     end for
10:    Divide $V_j^*$ into equivalence classes by the relation $\sim$. Remove the vectors in $V_j^*$
       such that each class contains only one vector.
11: end for
12: return Vector $v^* \in V_m^*$ which has the maximal product of its coordinates.
```

$v = (v_1, \dots, v_n) \in V_j$, there always exists a $v^* = (v_1^*, \dots, v_n^*) \in V_j^*$ such that $v_i^* \ge (1/\alpha^j) v_i$ for all i, $1 \le i \le n$. If $j = 1$, it is easy to see that $V_1^* = V_1$, hence the statement is obviously true. To prove the statement for j, assume that the statement is true for $j-1$. Consider the set V_j and an arbitrary vector $v = (v_1, \dots, v_n)$ of V_j. This vector v must be created in line 8 of Algorithm 2 from some vector $w = (w_1, \dots, w_n)$ in V_{j-1}.

Without loss of generality, we assume that v has the form of $(w_1 + s_{1j}, w_2, \dots, w_n)$ (note that $v_1 = w_1 + s_{1j}$ and $v_i = w_i$ for all $i = 2, \dots, n$). Using the inductive hypothesis above, there exists $w^* = (w_1^*, \dots, w_n^*) \in V_{j-1}^*$ such that $w_i^* \ge (1/\alpha^{j-1}) w_i$ for all i, $1 \le i \le n$. On the other hand, note that $w^* + s_{1j} \cdot e_1 = (w_1^* + s_{1j}, w_2^*, \dots, w_n^*)$ will also be created for V_j^* in line 8 of Algorithm 2, but it may be removed after line 10.

However, there is another vector $v^* = (v_1^*, \dots, v_n^*) \in V_j^*$ such that $v^* \sim (w^* + s_{1j} e_1)$. This yields

$$v_1^* \ge \frac{1}{\alpha} \cdot (w_1^* + s_{1j}) \ge \frac{1}{\alpha^j} \cdot w_1 + \frac{1}{\alpha} \cdot s_{1j} \ge \frac{1}{\alpha^j} \cdot (w_1 + s_{1j}) = \frac{1}{\alpha^j} \cdot v_1$$

and for i, $2 \le i \le n$, if $w_i^* \ne 0$, we have

$$v_i^* \ge \frac{1}{\alpha} \cdot w_i^* \ge \frac{1}{\alpha^j} \cdot w_i = \frac{1}{\alpha^j} \cdot v_i.$$

We now assume that Algorithm 1 returns a vector $v = (v_1, \dots, v_n) \in V_m$ such that the product $\prod_{i=1}^n v_i = OPT$ is maximal. Then, there must be a vector $v^* = (v_1^*, \dots, v_n^*) \in V_m^*$ such that $v_i^* \ge v_i/\alpha^m$ for all i, $1 \le i \le n$. This implies that

$$\prod_{i=1}^n v_i^* \ge \frac{1}{\alpha^{nm}} \prod_{i=1}^n v_i = \frac{1}{\alpha^{nm}} OPT.$$

Moreover, we have

$$\alpha^{nm} = \left(1 + \frac{\varepsilon}{2nm}\right)^{nm} \le e^{\varepsilon/2} \le 1 + \varepsilon.$$

The first inequality follows from the known inequality $(1+x/n)^n \leq e^x$ for all $n \geq 1$. The second inequality can be proven easily as follows. Consider function $f(x) = e^x - 1 - 2x$ in domain $x \in [0,1]$. The derivative $f'(x) = 0$ if and only if $x = \ln 2$. Therefore, we have $\max_{x\in[0,1]} f(x) = \max\{f(0), f(1), f(\ln 2)\} = f(0) = 0$.

Hence, we have

$$\prod_{i=1}^{n} v_i^* \geq \frac{1}{1+\varepsilon} OPT > (1-\varepsilon)OPT.$$

Let $M = (A,R,U)$ be a MARA-setting. We prove that Algorithm 2 has a running time that is polynomial in $|M|$ and $1/\varepsilon$, where $|M|$ denotes the size of M in some natural encoding. First, consider the set V_m^*, which has at most K^n vectors. Thus, the running time of the algorithm is in $\mathcal{O}(mK^n)$. On the other hand, we have

$$K = \lceil \log_\alpha B \rceil = \left\lceil \frac{\ln B}{\ln \alpha} \right\rceil = \left\lceil \frac{\ln B}{\ln\left(1+\dfrac{\varepsilon}{2nm}\right)} \right\rceil < \left\lceil \left(1+\frac{2nm}{\varepsilon}\right) \ln B \right\rceil.$$

The above inequality follows, since $f(a) = \ln a - 1 + 1/a$ is a continuous, increasing function on the interval $(1,\infty)$. This function is increasing on this interval, as $f'(a) = 1/a - 1/a^2 > 0$ for all $a > 1$. Hence, we have $f(a) > f(1) = 0$ for all $a > 1$. By choosing $a = \alpha$, the inequality follows.

Furthermore, note that $|M| \geq \log B = \log(e) \ln B$. Thus, we have

$$K \leq \left(1+\frac{2nm}{\varepsilon}\right) \frac{|M|}{\log(e)}$$

This proves the theorem for the maximum social welfare by the Nash product. Using the same algorithms, we obtain an FPTAS for maximum egalitarian social welfare with 1-additive utility functions. Indeed, assuming that Algorithm 1 returns the vector $v = (v_1,\dots,v_n) \in V_m$ such that the $\min\{v_1,\dots,v_n\} = OPT$ is maximal, Algorithm 2 must return a vector $v^* = (v_1^*,\dots,v_n^*) \in V_m^*$ such that $v_i^* \geq v_i/\alpha^m$ for all $i = 1,\dots,n$. Hence:

$$\min\{v_1^*,\dots,v_n^*\} \geq \min\left\{\frac{v_1}{\alpha^m},\dots,\frac{v_n}{\alpha^m}\right\} = \frac{1}{\alpha^m}\min\{v_1,\dots,v_n\} = \frac{1}{\alpha^m} OPT.$$

Choosing $\alpha = 1 + \varepsilon/2m$, we have

$$\alpha^m = \left(1+\frac{\varepsilon}{2m}\right)^m \leq e^{\varepsilon/2} \leq 1+\varepsilon.$$

and finally, $\min\{v_1^*,\dots,v_n^*\} \geq 1/(1+\varepsilon) OPT > (1-\varepsilon)OPT$. This proves the theorem. ❑

5. Conclusion

We have given new hardness results on egalitarian and Nash product social welfare optimization in multiagent resource allocation when utility functions are represented as straight-line programs. A new inapproximability result for Nash product social welfare optimization and FPTAS complement this picture. For future work, we propose the study of complexity and approximability of social welfare optimization problems for different representation forms and improving approximation algorithms. In particular, can we improve the hardness factor of $2/3$ and $1/2$ for the maximum Nash product social welfare and egalitarian social welfare problems with 1-additive utility functions? It is also very interesting to study whether or not these two problems are in APX (the class of problems allowing constant-factor approximation algorithms).

Acknowledgments: We thank the reviewers for their helpful comments.

References

[1] S. Arora and C. Lund. Hardness of approximations. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 10, pages 399–446. PWS Publishing Company, 1996.

[2] I. Bezáková and V. Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005.

[3] Y. Chevaleyre, P. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[4] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent resource allocation with *k*-additive utility functions. In *Proceedings of the DIMACS-LAMSADE Workshop on Computer Science and Decision Theory*, volume 3 of *Annales du LAMSADE*, pages 83–100, 2004.

[5] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent resource allocation in *k*-additive domains: Preference representation and complexity. *Annals of Operations Research*, 163:49–62, 2008.

[6] V. Conitzer, T. Sandholm, and P. Santi. Combinatorial auctions with *k*-wise dependent valuations. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 248–254. AAAI Press, 2005.

[7] P. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. *Artificial Intelligence*, 164(1–2):23–46, 2005.

[8] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[9] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.

[10] R. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131. ACM Press, 2004.

[11] T. Nguyen, M. Roos, and J. Rothe. A survey of approximability and inapproximability results for social welfare optimization in multiagent resource allocation. In *Website Proceedings of the Special Session on Computational Social Choice at the 12th International Symposium on Artificial Intelligence and Mathematics*, January 2012.

[12] N. Pippenger and M. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.

[13] S. Ramezani and U. Endriss. Nash social welfare in multiagent resource allocation. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, pages 117–131. Springer-Verlag *Lecture Notes in Business Information Processing #79*, 2010.

[14] M. Roos and J. Rothe. Complexity of social welfare optimization in multiagent resource allocation. In *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 641–648. IFAAMAS, May 2010.

[15] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1):116–127, 1976.

[16] C. Schnorr. The network complexity and the Turing machine complexity of finite functions. *Acta Informatica*, 7(1):95–107, 1976.

[17] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, second edition, 2003.