Approximability and Inapproximability of Social Welfare Optimization in Multiagent Resource Allocation

Inaugural-Dissertation

zur
Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Trung Thanh Nguyen
aus Hai Phong, Vietnam

Düsseldorf, im Juli 2013

Aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent:	Prof. Dr. Jörg Rothe Heinrich-Heine-Universität Düsseldorf
Koreferent:	Prof. Dr. Egon Wanke Heinrich-Heine-Universität Düsseldorf

Tag der mündlichen Prüfung: 21.11.2013

To my wife and my daughter

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation eigenständig und ohne unerlaubte Hilfe angefertigt und diese in der vorliegenden oder in ähnlicher Form noch bei keiner anderen Institution eingereicht habe.

Teile dieser Arbeit wurden bereits in den folgenden Schriften veröffentlicht bzw. zur Publikation angenommen: [NRR12, NNRR12b, NRR, NNRR12a, NNRR12c, NR13b, NR13a, NNRR, NR]. Andere Teile dieser Arbeit sind derzeit eingereicht: [BBL^{+}].

Düsseldorf, 18. Juli 2013

Trung Thanh Nguyen

Acknowledgments

This thesis is finished after three years of research at the Complexity Theory and Cryptology Group of the Department of Computer Science at the Heinrich Heine University of Düsseldorf.

I would like to thank first and foremost Prof. Jörg Rothe for introducing me to the field of Computational Social Choice, especially in Muliagent Resource Allocation, and for being a wonderful supervisor. I am very grateful to him for his great inspiration, patient guidance, and boundless support throughout my PhD study. His great knowledge of English also helped me so much in writing our articles. Without his supervision, I would never have had the chance to achieve the success in the scientific community. I would also like to thank Prof. Egon Wanke for being a referee for this thesis.

I also wish to express my deep gratitude to Prof. Robert Wisbauer for giving me a chance to study in Germany, a very beautiful country. His great encouragement and advice helped me a lot during the time of my study.

Many thanks also go to all my coauthors, including Jörg Rothe, Magnus Roos, Nhan-Tam Nguyen, Dorothea Baumeister, Jérôme Lang, Sylvain Bouveret and Abdallah Saffidine. I further thank all my colleagues at the Department of Computer Science of the Heinrich-Heine-Universität-Düsseldorf: Dorothea Baumeister, Claudia Forstinger, Guido Königstein, Anja Rey, Magnus Roos, Jörg Rothe and Lena Schend. Magnus Roos, thank you very much for sharing the room, thoughts, and good times. I am also thankful for my friends, Chan Khon Huynh and Thanh Phuc Luu, for proofreading parts of this thesis.

My research was supported by the Vietnamese Government under project 322. I also thank Deutscher Akademischer Austausch Dienst for the financial support for the German courses within three months in Vietnam and other two months in Germany.

I am deeply grateful to my parents for their continuous encouragement and support, to my wife and daughter who held me up with their endless love. This thesis is dedicated to them.

Abstract

Resource allocation is a fundamental problem in the field of multiagent systems, where a set of resources need to be allocated to a group of agents in a way so as to optimizing social welfare. In the last few years this problem has received much attention, especially due to its wide applicability domain (see the survey of Chevaleyre et al. [CDE$^+$06]). This thesis studies the approximability and inapproximability of several important optimization problems addressed in the area of multiagent resource allocation, focusing on two central representations of preferences, the bundle form and the k-additive form, and on various types of social welfare, ranging from quantitative measures (utilitarian, egalitarian and Nash product social welfare) to qualitative measures (envy-freeness).

First of all, we study the social welfare maximization problems, where we consider utilitarian and egalitarian social welfare and social welfare by the Nash product (for both total and average versions). We obtain several approximability and inapproximability results. For the utilitarian social welfare maximization with 2-additive utilities, we show that unless $\mathbf{P} = \mathbf{NP}$, it is impossible to achieve an approximation factor better than $^{21}/_{22}$. In addition, we prove that both egalitarian and Nash product social welfare maximization are **NP**-hard to approximate to within any factor, each for both the bundle form and the 3-additive form. For utility functions represented as additive form, we establish a hardness factor of $(^{8}/_{9}) + \varepsilon$ for total Nash product social welfare maximization and another hardness factor of $(^{2\sqrt{2}}/_{3}) + \varepsilon$ for the average version. Our positive results have all been achieved regarding the additive form. We provide a fast greedy approximation algorithm to within a factor of $(^{1}/_{m-n+1})$ for average Nash social welfare maximization, where n and m are the number of agents and the number of resources, respectively. We also prove that this problem admits a PTAS for the case of identical agents. Particularly, we obtain an FPTAS for maximization problems with respect to egalitarian and Nash product social welfare when the number of agents is not the part of input. Finally, we consider a special case when $n = m$ and present a polynomial-time algorithm that produces optimal solutions for both total and average Nash social welfare problems.

We then consider the problem of fairly distributing a number of indivisible goods among agents with additive utility functions, focusing on envy-freeness and its weaker notions. Instead of concentrating on envy-free allocations (which might not always exist), we seek to find an allocation with minimum envy. Based on a notion introduced by Chevaleyre et al. [CEEM07], we define several problems of minimizing the degree of envy and study their approximability.

We next concern the question of whether there exist truthful mechanisms for the problem of maximizing (total and average) Nash product social welfare and the problem of minimizing the degree of envy. We show that this is impossible for both exact and approximation mechanisms.

Finally we introduce a model of using the positional scoring rules for allocating indivisible resources, where agents express their preferences only by ranking single resources from the most preferred to the least preferred. We then define several problems, either in their decision or optimization version, and study them from a computational point of view.

Zusammenfassung

Das Aufteilen von Gütern ist ein fundamentales Problem im Bereich der Multiagenten-Systeme, bei dem eine Menge von Gütern unter einer Gruppe von Agenten aufgeteilt werden soll, sodass die soziale Wohlfahrt dabei optimiert wird. In den letzten Jahren wurde diesem Problem viel Aufmerksamkeit gewidmet, gerade weil es viele Anwendungen hat (siehe die Übersicht von Chevaleyre et al. [CDE+06])). Diese Arbeit behandelt die Approximierbarkeit und die Unapproximierbarkeit von verschiedenen, wichtigen Optimierungsproblemen im Bereich der Güteraufteilung mit mehreren Agenten, wobei der Fokus auf zwei zentralen Formen liegt, die Präferenzen der Agenten zu repräsentieren: die Bündel-Form und die k-additive Form. Weiterhin werden verschiedene Formen der sozialen Wohlfahrt untersucht, diese reichen von quantitativen Maßen (utilitarische, egalitarische und Nash-Produkt soziale Wohlfahrt) bis hin zu qualitativen Maßen (Neidfreiheit).

Am Anfang widmen wir uns Probleme, die die soziale Wohlfahrt maximieren, dabei behandeln wir utilitarische und egalitarische soziale Wohlfahrt und soziale Wohlfahrt in Hinblick auf das Nash-Produkt (für die totale und die durchschnittliche Version). Wir erhalten verschiedene Approximierbarkeits- und Unapproximierbarkeits-Resultate. Für utilitarische soziale Wohlfahrt mit 2-additiven Nützlichkeitsfunktionen zeigen wir, dass es unmöglich ist einen Approximationsfaktor besser als $^{21}/_{22}$ zu erhalten, solange nicht $\mathbf{P} = \mathbf{NP}$ gilt. Weiterhin zeigen wir, dass die egalitarische soziale Wohlfahrt und die soziale Wohlfahrt in Hinblick auf das Nash-Produkt für jeden Faktor **NP**-hart zu approximieren sind. Beides gilt sowohl für die Bündel-Form als auch für die 3-additive Form. Sind die Nützlichkeitsfunktionen additiv gegeben, erhalten wir einen Härtefaktor von $(^8/_9) + \varepsilon$ für die Optimierung der totalen sozialen Wohlfahrt in Hinblick auf das Nash-Produkt und einen weiteren Härtefaktor von $(^{2\sqrt{2}}/_3) + \varepsilon$ für die durchschnittliche Version. Alle unseren positiven Ergebnisse werden für die additive Form erreicht. Wir geben einen schnellen Greedy-Algorithmus an, der ein Ergebnis innerhalb eines Faktors von $^1/_{(m-n+1)}$ für die Optimierung der durchschnittlichen sozialen Wohlfahrt in Hinblick auf das Nash-Produkt liefert. Dabei ist n die Anzahl der Agenten und m die Anzahl der Güter. Ausserdem beweisen wir, dass es für den Fall von identischen Agenten ein

PTAS gibt. Genau genommen erhalten wir ein FPTAS für die Optimierung der egalitarischen sozialen Wohlfahrt und derer mit Hinblick auf das Nash-Produkt, falls die Anzal der Agenten nicht Teil der Eingabe ist. Am Ende betrachten wir den Spezialfall, dass $n = m$ gilt und wir geben einen Polynomialzeit-Algorithmus an, der die Optimallösung sowohl für das totale als auch für das durchschnittliche Problem die soziale Wohlfahrt in Hinblick auf das Nash-Produkt zu optimieren liefert.

Dann betrachten wir das Problem, eine Menge von unteilbaren Gütern zwischen Agenten mit additiven Mützlichkeitsfunktionen aufzuteilen, wobei auf Neidfreiheit und ihre schwächeren Bedeutungen geachtet wird. Anstatt uns auf neidfreie Aufteilungen zu konzentrieren (die nicht immer existieren), versuchen wir Aufteilungen zu finden, bei denen der Neid minimiert wird. Basierend auf Begriffen, die Chevaylere u.a. [CEEM07] eingeführt haben, definieren wir verschiedene Probleme um den Grad des Neides zu minimieren und betrachten deren Approximierbarkeit.

Dann widmen mir uns der Frage, ob es einen wahrhaftigen Mechanismus gibt, um die (totale und durchschnittliche) soziale Wohlfahrt in Hinblick auf das Nash-Product zu maximieren, sowie dem Problem den Grad des Neides zu minimieren. Wir beweisen, dass dies sowohl für exakte als auch für approximative Mechanismen unmöglich ist.

Zum Schluss führen wir ein Modell ein, dass es uns erlaubt, positionelle Punktregeln zum Aufteilen von unteilbaren Gütern zu nutzen, wobei die Agenten ihre Präferenzen nur dadurch zum Ausdruck bringen, dass sie die einzelnen Güter nach ihrer Wertigkeit anordnen, von wertvoll bis wertlos. Wir definieren dann verschiedene Probleme, entweder in der Entscheidungsversion oder in der Optimierungsversion, und betrachten sie aus der Sicht der Berechenbarkeit.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Resource allocation, which deals with distributing a number of resources (financial, foods, energy, etc.) amongst a group of agents, is a central task in the field of multiagent systems (MAS). Much work so far in the area of MAS has been either directed towards providing mechanisms for resource allocation (e.g, combinatorial auctions), or preprocessed by a resource allocation mechanism (e.g, collaborative problem solving). Moreover, a lot of important application areas concerned in the MAS research community are also relevant to multiagent resource allocation (e.g, electronic commerce).

A typical example of resource allocation is a (combinatorial) auction where a number of (discrete) items are auctioned concurrently among potential bidders (agents). Each bidder can place bids on (combinations of) items. The usual goal of an auctioneer is to provide a mechanism for finding an allocation that maximizes the revenue from the sale of items. Auction theory has been widely studied in economics of the last forty years. Nowadays, its applications can be found in various areas of real life, such as spectrum license (MacMillan [Mac94]), allocation of airport take-off and landing time slots (Rassenti et al. [RSB82]), sales of online seats (Eso [Eso01]), electricity markets (Ausubel and Cramton [AC04]), and trading (Abrache et al. [ACG05]).

As another example, we consider the following situation. A country X receives one million dollars in donation for the purpose of developing three main areas: healthcare, education and housing. Generally, the distribution of this funds among these three areas can be considered on the three levels of decision making. At the first level, one can consider how to allocate the spending to three areas. Should the funding be divided equally among those or should one program get all the money? At the second level, assume that each program gets a portion of the one million dollars, the next decision in the education area is how to best allocate the funds among various education interests. Should most of the funds be dedicated to building schools or to buying educational equipment? Or, should one spend a part of the funds to grant scholarships to pupils who would not normally be able to attend the school? The last level of decision making involves allocating the financial resources among individuals. Depending on the policies of the country, the decisions at this level may include: who will get the next scholarship when there are many pupils in the list? How should one distribute the scholarships in a way that ensures fairness?

The main issue addressed in multiagent resource allocation is to model mathematically the problems of allocating resources (either as decision or as optimization problems) and then looking for the methods for solving them efficiently. For the last decade much of the effort has been devoted to study the computational complexity of various decision problems arising in the context of multiagent resource allocation. Many of the problems have been shown to be **NP**-hard in general and it seems unlikely that one can develop exact algorithms that can produces optimal solutions in polynomial time. Therefore, it is of great interest to know whether one can design an "approximation algorithm" - an efficient algorithm that is guaranteed to produce a good approximation to the optimum solution.

The dissertation obtains both approximation algorithms and lower bound results for approximating social welfare optimization problems in multiagent resource allocation. Here, we present a brief summary of the main contributions of this dissertation.

1. We provide the first lower bounds on the approximability of several optimization problems where the languages used to represent the preferences of agents over the alternative bundles of resources are the *bundle form* and the *k-additive form*. These problems include: MAXIMUM UTILITARIAN SOCIAL WELFARE, MAXIMUM EGALITARIAN SOCIAL WELFARE, MAXIMUM TOTAL NASH SOCIAL WELFARE, and MAXIMUM AVERAGE NASH SOCIAL WELFARE. To the best of our knowledge, no hardness results for these problems have been known so far, except the case of maximizing utilitarian social welfare with the bundle form, which was given by Lehmann et al. [LOS99, LOS02], and another case of maximizing egalitarian social welfare with the additive form, which was obtained by Bezáková and Dani [BD05].

2. We give the upper bounds for approximating MAXIMUM EGALITARIAN SOCIAL WELFARE, MAXIMUM TOTAL NASH SOCIAL WELFARE, and MAXIMUM AVERAGE NASH SOCIAL WELFARE with respect to the additive form. Namely, we propose a first approximation algorithm for the two latter problems. In particular, we introduce a deterministic algorithm based on dynamic programming, which can

be used to construct fully polynomial-time approximation schemes for every social welfare maximization problems, when the number of agents is fixed.

3. We consider a variant of the social welfare maximization problem where the number of agents and the number of resources are the same, and provide an efficient polynomial-time exact algorithm for solving it.

4. We introduce several variants of the problem of finding envy-free allocations obtained by relaxing envy-freeness by the notion of degree of envy, and then present some approximability results for them.

5. We introduce a method of using the positional scoring rules for allocating indivisible resources, and we model and study several problems in this context from the computational point of view.

This dissertation is organized as follows. Chapter 2 provides the fundamentals of the field of multiagent resource allocation and a short introduction to the computational complexity theory. In particular, we introduce the basic methods for designing approximation algorithms and the main techniques for proving the hardness of approximation.

In Chapter 3, the first section is devoted to give the basic notions of social welfare as well as of two types of representation of utility functions, the bundle form and the k-additive form. We will give a brief overview of known results in Section 3.2. In Sections 3.3, we provide lower bounds and upper bounds on the approximability of the social welfare maximization problems. This chapter concludes with a summary of the obtained results and a list of open questions in Section 3.4.

Next, in Chapter 4 we concentrate on relaxing the notion of envy-freeness and study the problem of finding allocations that minimize the envy.

Chapter 5 deals with the inapproximabity results for truthful mechanisms for the problem of maximizing Nash product social welfare and the problem of minimizing the envy.

Finally, in Chapter 6 we consider the problem of allocating indivisible resources using positional scoring rules. Section 6.1 will be devoted to present the basic notions of positional scoring rules that was originally studied in voting theory. The formal problem definitions are given in Section 6.2. Section 6.3 presents the main results related to computational complexity and approximation algorithms. This chapter concludes with a summary and a discussion for future work in Section 6.4.

Chapter 2

Preliminaries

In this chapter we will introduce briefly the basics of the field of multiagent resource allocation which will be discussed in this thesis. Furthermore, we also review relevant background knowledge of computational complexity theory and approximation algorithm theory.

2.1 Multiagent Resource Allocation

Multiagent resource allocation (MARA) is a central topic in both computer science (artificial intelligence) and economics (social choice theory), that deals with providing mechanism to allocate a number of resources amongst two or more agents. While economics concerns the quality of allocations, the computational aspect is the main issue studied in computer science. Therefore, the field of MARA has interdisciplinary characteristics and relates to a wide range of applications, such as auction, scheduling, network routing, airport traffic management, allocation of mineral riches in the ocean bed, the fair and efficient exploitation of earth observation satellites, greenhouse gas emissions reduction, divorces, etc. One may ask, what exactly is MARA? The following definition of MARA introduced in [CDE+06] is tentative and perhaps not as precise as should be desired.

"*Multiagent Resource Allocation is the process of distributing a number of items amongst a number of agents.*"

In what follows, we will give a short overview of important parameters involved in a MARA system as well as main research topics. For further details, we refer to the survey paper of Chevaleyre et al. [CDE+06] and the references therein.

Agents

An agent is an autonomous decision maker that may have different preferences over the alternatives (allocations). For example, an agent can be a person or a machine. The term multiagent system is used to refer to a group of agents, which interact or work together in order to perform some tasks or achieve some goals.

Resources

In the context of MARA, resources (sometimes called *goods, objects* or *items*) can be food, land, oil, energy or anything else which can be considered valuable. For the purpose of study, we can categorize resources according to their properties and characteristics. For example, we can distinguish between *divisible* and *indivisible* resources. Generally, divisible resources are things that can be divided into smaller parts without lost of value, such as a cake, while an indivisible resource, such as a car, can only be assigned to a single agent in its entirety. We also discriminate between *continuous* and *discrete* resources. A continuous resource (e.g, fuel, liquid, energy) could be divided into small parts, each can be seen as an indivisible (and thus discrete) resource, before being allocated to agents. Alternatively, resources can be treated as *static* or not. Non-static resources are either consumable (e.g, fuel) or perishable (e.g, food), whereas static resources do not change their properties during the process of allocating. Another distinction is between *sharable* and *non-sharable* resources. An example of sharable resources are internet servers that can be allocated to more agents at once. Finally, we can distinguish between *single-unit* and *multi-unit* (where multiple copies of many different items are available, e.g, a dozen bottles of beer).

The survey of Chevaleyre et al. [CDE$^+$06] provides a distinction between the types of resources in more detail. In this thesis, we only consider the scenarios where resources are static, single-unit, indivisible and that cannot be shared. We also use the terms good, object and item synonymously instead of resource.

Allocation

In general, an allocation is a particular assignment of resources among agents. An allocation is said to be complete if every resources are assigned completely to agents, otherwise it is called an incomplete allocation. If an allocation is not specifically referred to as being incomplete, then it is understood to be complete.

Preferences

An important parameter of a MARA is the preference representation of agents over resources. Depending on the information provided by agents, in part or complete, precise or vague, the preferences can be modeled in two main ways: *cardinal preference* and *ordinal preference*.

Cardinal Preference

Much of study in economics, especially in *decision making*, or in social choice theory (e.g, *voting*), relies on the idea that agents' preferences can be expressed in certain numeric manner. In a bit formal, each agent is equipped with a so-called *utility function*, which is a mapping from a set of subsets of resources to a numerical set $\mathbb{F}$ (such as the set $\mathbb{N}$ of nonnegative integers, the set $\mathbb{Z}$ of integers, the set $\mathbb{Q}$ of rational numbers, and the set $\mathbb{Q}^+$ of nonnegative rational numbers). Utility functions allow agents to evaluate precisely the utility of the bundles of resources they received. The idea behind utility functions mapping bundles of resources rather than single resources to values in $\mathbb{F}$ is that agents might be willing to pay either more or less for a bundle than the sum of their utilities for this bundle's single items. For example, owning a pair of matching shoes is likely to be more valuable to an agent than the sum of the values each single shoe has for this agent. On the other hand, an agent who is willing to bid on 100 identical items might expect some discount and so has less utility for the bundle of 100 items than 100 times the utility assigned to a single item.

There are several choices for representing utility functions in the way of cardinal preference, each of them has advantages and disadvantages under different scenarios.

Bundle form: In this form, agents enumerate all possible bundles of resources to which they assign a non-zero value. This is perhaps the most intuitive representation of utility functions, but its drawback is that it takes exponential time in the number of resources in the worst case for the description. For instance, given a set of only 10 single resources, it may require 2^{10} real numbers to present each possible agent's preference.

k-additive form: This form of representation is inspired from the notion of k-additive functions that was first used in fuzzy measure theory [Gra97] and was then introduced in the context of combinatorial auction by Conitzer et al. [CSS05]. For a fixed positive integer k, a utility function is said to be k-additive if the utility of every bundle of resources B is represented as the sum of utilities assigned to subsets of B with cardinality

less than or equal to k. A formal definition of the k-additive form will be provided in Section 3.1.

Straight-line program: 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 B and the output as the binary representation of $u(B)$, we can say that C (or a corresponding straight-line program) represents utility function u. The formal definition of this kind of representation can be found in [DWL05].

Bidding language: The bidding language has been developed to encode bids more succinctly in combinatorial auction. Typically, it is represented as a set of atomic bids, each of the form (B, p) in which p is maximum price that a bidder is willing to pay for the bundle B. Two main bidding languages are the OR and XOR languages. The OR language is defined as follow: given a bid of the form $(B_1, p_1)\,\text{OR}\ldots\text{OR}\,(B_k, p_k)$, the value of the bundle B is determined by:

$$\max_{I} \sum_{i \in I} p_i$$

where $I \subseteq \{1, \ldots k\}$ such that for all $i \neq j \in I$, $B_i \cap B_j = \emptyset$.

One can show that the OR language is not fully expressive since it cannot represent the bids that are subadditive.

The XOR language is more expressive than the OR language: given a bid of the form $(B_1, p_1)\,\text{XOR}\ldots\text{XOR}\,(B_k, p_k)$, the value of the bundle B is defined to be

$$\max_{i, B_i \subseteq B} p_i$$

It is easy to see that the XOR language can be used to represent any utility function but it is less compact than the OR language in some cases. For example, any additive function on m items can be represented by OR bids of size m while the use of XOR requires 2^m bids. For a detailed discussion of the bidding languages OR, XOR as well as the combination of these two languages, OR-of-XOR and XOR-of-OR, we refer to the bookchapter of Nisan [CSS06].

Ordinal preference

For this type of representation, agents express their preferences over bundles of resources via binary relations, denoted by $\succeq$, which are reflexive and transitive but not necessarily complete (see Example 2.1). For example, the relation $A \succeq B$ expresses that agent prefers the bundle A at least as much as the bundle B. If $A \succeq B$ and $B \succeq C$ then $A \succeq C$.

Example 2.1 *Consider an example with two agents $\{a_1, a_2\}$ and three resources $\{a, b, c\}$ and the agents' preferences are given in Table 2.1. The preference of agent a_1 is complete, but the preference of agent a_2 is not.*

Table 2.1: Utilities of the agents for Example 2.1.

Agents	Preferences
a_1	$\{a,b,c\} \succeq \{a,b\} \succeq \{a,c\} \succeq \{a\} \succeq \{b,c\} \succeq \{b\} \succeq \{c\}$
a_2	$\{a\} \succeq \{b\} \succeq \{c\}$

Unlike the cardinal preference, ordinal representations of preferences emphasize the order of positions of the alternatives (bundles of resources). In many situations, it would be difficult to have preferences of agents accurately: someone can say that she prefers coke to sprite but if we ask her to map her ranking to some cardinal scale, she would be confused. In such situations, obtaining an ordinal preference seems to be much more easier than a cardinal one.

In fact, any cardinal preference induces an ordinal preference but the reverse is not true. Indeed, if we know the cardinal preference of an agent represented by a function $u : 2^R \rightarrow \mathbb{F}$, then her ordinal preference over 2^R is determined by the relation $\succeq$ as follows: for any two bundle of resources $A, B \subseteq 2^R$: $A \succeq B$ if and only if $u(A) \geq u(B)$. Conversely, if we know the ranking of agents over the bundles of resources, it is not clear that whether we know the cardinal intervals between the two bundles A and B. This situation is the same as in an athletic competition where if we only know the athletes placing first, second, third and so on, we cannot say that the winner beat the second one by precisely as much time as the second athlete beat the third one.

Social Welfare

Social welfare is one of the most important parameters in a MARA. This notion is inspired from the works in welfare economics and social choice theory. Intuitively, by means of social welfare, it shows how individual preferences can be aggregated into desirable social or collective decisions. For example, assume that agents express their preferences by using utility functions, every given allocation of resources to agents induces a vector of utilities (*collective utility*) that can be aggregated to a single value, the social welfare of this allocation. There are different concepts of social welfare, relying on both quantitative measure, including *utilitarianism, egalitarianism, elitistism, k-rank dictator, leximin ordering, Nash product,* and qualitative measure such as *Pareto efficiency, envy-freeness, proportional.*

utilitarianism: *utilitarian social welfare* sums up the agents' individual utilities in a given allocation, thus providing a useful measure of the overall—and also of the average—benefit for society. For instance, in a combinatorial auction the auctioneer's aim is to maximize the auction's revenue (i.e., the sum of the prizes paid for the items auctioned), no matter which agent can realize which utility.

egalitarianism: *egalitarian social welfare* (also known as *Rawlsian* measure) gives the utility of the agent who is worst off in a given allocation, which provides a useful measure of fairness in cases where the minimum needs of all agents are to be satisfied. For example, think of distributing humanitarian aid items (such as food, medical aid, blankets, tents, etc.) among the needy population in a disaster area (e.g., an area hit by an earthquake or a tsunami). Guaranteeing every survivor's continuing survival is the primary goal in such a scenario, and it is best captured by the notion of egalitarian social welfare.

elitistism: The elitist social welfare coincides with the maximal individual utility in the system and thus in some sense it is not a fair measure of social welfare. However, it may be used in several distributed computing applications where the system designer only concerns one of agents who can achieve the utility as much as possible.

k-rank dictator: This measure is defined as the utility of the k-th happiest agent in an allocation. One can see that the egalitarian social welfare is exactly the 1-rank dictator while the elitist social welfare is corresponding to the n-rank dictator, where n is denoted by the number of agents. Thus, both egalitarianism and elitistism are specific cases of the k-rank dictator. A special case when $k = \lceil n/2 \rceil$ is called the *median*-rank dictator.

leximin ordering: This is a refinement of egalitarianism (and thus elitistism and k-rank dictator) that, informally, works by comparing first the utilities of the least satisfied agents, and when these coincide, compares the utilities of the next least satisfied agents, and so on. Stated differently, the leximin ordering is a lexicographical ordering over the ordered utility vectors (an ordered utility vector is a vector in which its coordinates are arranged in an increasing order) derived from the allocations. As well as the egalitarian social welfare, it favors fair division of resources.

Nash product: The *Nash product social welfare*, which is inspired from the work of John Nash [Joh50], is the product of individual agent utilities. It can be seen as a compromise between utilitarian and egalitarian social welfare. On the one hand, it has the (strict) monotonicity property of utilitarian social welfare because an increase in any agent's utility leads to an increase of the Nash product (provided all agents have positive utility). On the other hand, the Nash product increases as well when reducing inequitableness among agents by redistributing utilities, thereby providing a measure of fairness. Looking at the ordering that is induced by the allocations, the "social welfare ordering," Moulin [Mou04] presents further beneficial properties of the Nash product. For example, the Nash product is uniquely characterized by independence of individual scale of utilities,[1] i.e., even if different "currencies" are used to measure the agents' utilities, the social welfare ordering remains unaffected.

Pareto efficiency: An allocation is Pareto-efficient or Pareto-optimal if no agent can improve her welfare without reducing other agents' welfare. This idea goes back to Vilfredo Pareto (Italian economist, 1848-1923), and generally is viewed as the most fundamental criterion for efficiency.

envy-freeness: The *envy-freeness* is perhaps one of the most important notions of fairness in the context of resource allocation. An allocation is said to be envy-free if no agent wants to exchange his bundle of resources with that of any other agent's. Unfortunately, such an allocation might not exist in many cases because of the requirement that all resources must be assigned completely to agents. For example, consider an instance with two agents and only one resource. We can not have an allocation that ensures envy-freeness. The reason is that the resource is only assigned to one of the two agents and if so, the other agent would be envious.

[1]Similarly, utilitarian social welfare is characterized by independence of individual zeros of utilities: A constant shift of an agent's utility function does not change the social welfare ordering.

But if both agents get nothing (the empty bundle), the allocation is envy-free. Instead of finding envy-free allocations, we may seek to compute allocations such that the *envy* is as small as possible (see [LMMS04]).

proportional: An allocation is proportional if each agent thinks that she got at least $1/n$-th of the resources (where n is the number of agents). It is known that if an allocation is envy-free, then it must also be proportional, but the converse is not true. An example of using this measure is the problem *cake-cutting* (see Brams and Taylor [BT96]).

Allocation Procedure

A resource allocation procedure is a procedure for achieving a certain desirable outcome, being either optimal or feasible. Typically, such a procedure could be *centralised*, where a single entity plays the role deciding what should be the final allocation; or *distributed*, where the final allocation is reached after a sequence of negotiation steps for which agents agree on exchanging resources. Combinatorial auction is a typical example of centralised approach in which the bidders place their bids on combinations of items and the auctioneer, who plays the role as the central entity, will decide the result of the auction. For the basic introduction as well as a survey of results so far in this area, we refer to the textbooks of Peter Cramton, Yoav Shoham, and Richard Steinberg [CSS06] and a bookchapter of Blumrosen and N. Nisan [BN07].

The most common approaches for the distributed allocation are *negotiation protocols*. These protocols allow agents to negotiate in a group of two agents (*bilateral*) or between any number of agents(*multilateral*), in order to agree on deals to exchange the resources. There are several protocols that have been developed for negotiation over the resources and perhaps, among those, the most popular one is *Contract-Net* protocol which is introduced in [Smi80]. Other extensions of this protocol can be found in the papers of Sandholm [San93], Golfarelli et al. [GMR97] and Sousa et al. [SRN03].

Both the centralised and distributed approaches for multiagent resource allocation have their own advantages and disadvantages. Centralised procedures are known to be more popular in practice since they do not require much communication between agents to implement such procedures. Furthermore, one can design the powerful algorithms for finding the desirable allocations. Nevertheless, the drawback of this approach is the limit of computational resources. That means it is unlikely to implement when the number of agents as well as the number of resources increases. Hence, one of the most important issues is determining the complexity of computing an optimal allocation via centralised protocols? In some situations, agents might not want to report their truth preferences to the auctioneer, and thus a centralised protocol cannot be applied. The disadvantages of the distributed approach lie on the quality of an allocation that it found. However, it might be a good choice for the case where computing optimal allocations is hard. In this thesis, we concentrate on the first type of allocation procedure.

Our Research Topics

The survey of Chevaleyre et al. [CDE$^+$06] enumerates the main topics involved in the area of multiagent resource allocation. Among those, we are interested in studying the computational aspects of social welfare optimization problems. Since most of them are **NP**-hard to solve, we are looking for the methods that can be used for solving these problems, exactly or approximately, or deriving approximation hardness results.

2.2 Computational Complexity

Landau's symbols

In complexity theory, Landau's symbols are used to characterize how functions compare asymptotically. In particular, it allows us to formulate the upper and lower bounds on a function. Formally, given the functions: $f, g : \mathbb{N} \to \mathbb{R}^+$, we define:

- **asymptotic upper bound:**

  $f \in \mathcal{O}(g) \Leftrightarrow \exists c > 0, \exists n_0, \forall n \geq n_0 : f(n) \leq c \cdot g(n)$, (f asymptotically grows *at most as fast as* g).

- **asymptotic lower bound:**

  $f \in \Omega(g) \Leftrightarrow \exists c > 0, \exists n_0, \forall n \geq n_0 : f(n) \geq c \cdot g(n)$, (f asymptotically grows *at least as fast as* g).

- **asymptotically tight bound:**

  $f \in \Theta(g) \Leftrightarrow f \in \mathcal{O}(g) \cap \Omega(g)$, (f asymptotically grows *just as fast as* g).

Table 2.2: Some typical upper bounds on the time complexity

Name	Running time
constant	$\mathcal{O}(1)$
logarithmic	$\mathcal{O}(\log n)$
linear	$\mathcal{O}(n)$
quasilinear	$\mathcal{O}(n \log n)$
polynomial	$\mathcal{O}(n^c), c > 1$
exponential	$\mathcal{O}(c^n), c > 1$
factorial	$\mathcal{O}(n!)$
super-exponential	$\mathcal{O}(n^n)$

By using the symbols $\mathcal{O}$ and Ω, we can give upper and lower bounds, respectively, for the asymptotic running time of an algorithm. For example, a linear search algorithm for the problem of finding a biggest element in an unordered array of n numbers has a running time of $\mathcal{O}(n)$. As another example, the lower bound on the best-case running time of the insertion sort algorithm is $\Omega(n)$ (see [CLRS09]). Finally, the merge sort, a standard algorithm for sorting an array of n arbitrary elements, has an upper bound of $\mathcal{O}(n \log(n))$ and a lower bound of $\Omega(n \log(n))$ on the running time. Thus, the asymptotically exact time complexity of this algorithm is $\Theta(n \log(n))$.

Complexity classes

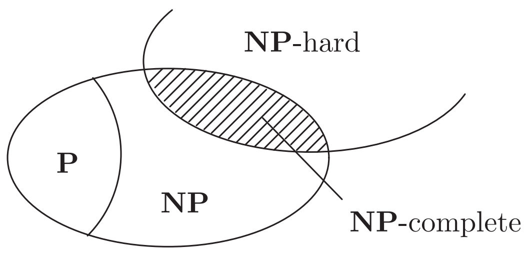
Many computational problems are known to be tractable in the sense that they can be solved in polynomial time (such as *primality testing* (PRIMES)). Unfortunately, many other important problems are *hard* to solve, in the sense that no polynomial algorithms are known for them (such as *traveling salesman problem (TSP)*). The field of computational complexity deals with classifying computational problems according to their inherent difficulty, that is, the amount of *computational resources* needed to solve the problems. There are two central types of computational resources: *time* and *space*, and both depend on the size of input. Time complexity estimates the number of steps to solve the problem, while space complexity measures the number of memory cells that are needed to solve the problem. Landau's symbols can be used to describe the upper and lower bound of time and space complexities. Table 2.2 lists some typical upper bounds on the time complexity.

In general, there are at least four categories that a computational problem could fall into, including: *decision problem, search problem, counting problem* and *optimization problem.* In this thesis, we focus on decision problems and optimization problems.

A decision problem is a problem whose answer is *yes* or *no* for every input. For example, Primes is the decision problem of determining whether or not a given integer N is prime. Another example is TSP which asks whether a graph has a Hamiltonian cycle (i.e., a cycle which visits every vertex exactly once). Decision problems can be classified into complexity classes according to their resource consumption (the resource is time or space). The two best known complexity classes are **P** and **NP**. **P** (the abbreviation **P** stands for polynomial time) is the class of all decision problems that can be solved by a *deterministic Turing machine* in time polynomial in the length of its input. The Turing machine is a simple abstract model of computation which is used in modern computability and complexity theory. A comprehensive description of it can be found in [GJ79]. To show that a problem is in **P**, one has to provide a polynomial-time algorithm that runs on every input of the problem. For instance, Primes was shown to be in **P** by using a so-called AKS primality test algorithm (a.k.a Agrawal-Kayal-Saxena primality test, see [AKS02]). The class **NP** (the abbreviation **NP** refers to nondeterministic polynomial time) consists of decision problems which are solvable by a *nondeterministic Turing machine* in polynomial time. Equivalently, a decision problem A is in **NP** if every yes-instance of A has a short certificate that is verifiable in polynomial time. For example, TSP is in **NP** because given a path in the graph, one can verify easily if it is a Hamiltonian cycle. The nondeterministic Turing machine can find such a cycle in n steps, where n is the number of vertices of the graph, by guessing which vertex we should visit to next at every step. However, we do not know if this can be done by a deterministic Turing machine in polynomial time. In fact, it is easy to show that $\mathbf{P} \subseteq \mathbf{NP}$ but the converse is not known to hold. This is probably the most important open question in computer science and mathematics. It is widely believed that $\mathbf{P} \neq \mathbf{NP}$ but nobody has been able to prove this.

By the assumption that $\mathbf{P} \neq \mathbf{NP}$, there are problems that are in **NP** but not in **P**. One can classify decision problems within **NP** according to how hard they are. The class **P** contains the "easiest" problems which can be solve in polynomial time while **NP**-complete is the class of "hardest" problems in **NP**. By the 'hardest" problems, we mean that if we can prove that even one **NP**-complete problem is polynomially solvable, then every **NP**-complete problems are polynomially solvable and thus it implies that $\mathbf{P} = \mathbf{NP}$. To understand what exactly **NP**-complete problems are, we need to use a so-called *polynomial-time many-one reduction* that is defined as follows.

Figure 2.1: The relationship between **NP**, **NP**-complete, **NP**-hard and **P**, assuming $\mathbf{P} \neq \mathbf{NP}$.



Definition 2.1 *A decision problem A is polynomial-time many-one reducible to a decision problem B, denoted by $A \leq_m^p B$, if and only if there is a polynomial-time function f such that, for any instance I of problem A, the answer of problem A for this instance is yes if and only if the answer of problem B for instance $f(I)$ is yes.*

Obviously, if $A \leq_m^p B$ and B is in **P** then A is in **P** too. Indeed, in order to solve an instance I of A, we transform (in polynomial time) I into an instance $f(I)$ of B and then solve it. If the problem A has a property that any problem $B \in \mathbf{NP}$ is reducible to A in polynomial time then A is called **NP**-hard. Intuitively, a problem is **NP**-hard if it is at least as hard as any problem in **NP**. Note that a **NP**-hard problem might not be in **NP**, for example, the Halting problem which determines whether a particular program or an algorithm will terminate or run forever. If A is **NP**-hard and $A \in \mathbf{NP}$ then A is **NP**-complete. The relationship between **NP**, **NP**-complete, **NP**-hard and the class of problems whose optimal solution can be found in polynomial time, **P**, is illustrated in Figure 2.2.

In computational practice, when we want to show that a given problem C is **NP**-hard, we need only to take a problem B which is already known to be **NP**-hard and prove that B can be polynomially reduced to C. Since the polynomial reduction is transitive, any problem $A \in \mathbf{NP}$ can be also polynomially reduced to C and thus, C is **NP**-hard. For example, we consider a simple allocation problem as follows. The *Santa Claus* has m packs of candies, each has a_i candies, and he wish to allocate them among n children without opening the packs. The question is if there is an assignment such that the least lucky child is as happy as possible. This problem is shown to be **NP**-hard, even with only two children, by a reduction from a well-known **NP**-hard problem Partition which asks, given a set of positive integer numbers $S = \{a_1, \dots, a_m\}$, whether there exists a subsets $C \subset S$ such that the sum of the elements of C equals to the sum of the elements of $S \setminus C$.

There are other complexity classes that now exist in computer science beyond the **NP** such as $\mathbf{DP}, \mathbf{P^{NP}}, \mathbf{NP^{NP}}$. Since we will not go further in using these classes in this thesis, we do not define them here but refer to the textbooks [Pap95, Rot05].

Optimization problems

After introducing decision problems and the two important complexity classes, **P** and **NP**, we now turn to optimization problems. We will introduce the classes **PO** and **NPO** of optimization problems that are counterparts of the classes **P** and **NP**, respectively, for decision problems. We start with a formal definition of an optimization problem.

Definition 2.2 *An optimization problem* Π *is characterized by the four components* (I, SOL, val, OPT)*, where* I *is the set of instances of* Π *and for each instance* $x \in I$*,*

- $SOL(x)$ *is the set of all feasible solutions to* x*,*
- *For each solution* $s \in SOL(x)$*,* $val(s)$ *denotes the (positive) value of* s*,*
- $OPT(x)$ *is the optimal value of the instance* x*;*

There are two major types of optimization problems:

- *maximization* problem where $OPT(x) = \max_{s \in SOL(x)}\{val(s)\}$, and
- *minimization* problem where $OPT(x) = \min_{s \in SOL(x)}\{val(s)\}$

Let us take two examples of optimization problem as follows.

Maximum-2-Satisfiability (Max-2-Sat)
Instance: A boolean formula φ in conjunctive normal form consisting of clauses having two literals each.
Solution: A truth assignment to the variables of φ.
val: The number of satisfied clauses by a truth assignment.
OPT: The maximal possible number of simultaneously satisfiable clauses of φ.

Minimum Traveling Salesman Problem (TSP)
Instance: A complete undirected graph G that has a nonnegative integer weight associated with each edge.

Solution: A Hamiltonian cycle.
val: The weight of a Hamiltonian cycle.
OPT: The minimal possible weight of a Hamiltonian cycle.

In fact, any optimization problem $\Pi = (I, SOL, val, OPT)$ has an associated decision problem Π' which asks, for some nonnegative number k, whether or not there exists a solution s of instance $x \in I$ such that $val(s) \geq k$ if Π is a maximization problem, or $val(s) \leq k$, if Π is a minimization problem. For example, the decision version of the maximization problem Max-2-Sat above can be stated as follows: given a boolean formula φ in conjunctive normal form consisting of clauses having two literals each and a positive integer number k, the question is if there exists a truth assignment to the variables of φ such that the number of satisfied clauses is greater than or equal to k. Clearly, Π' is not harder than Π. In the other words, if an optimization problem can be solved in polynomial time, then its corresponding decision version can be polynomially solved.

In what follows, we focus on optimization problems whose decision versions are in **NP**. The formal definition is as below.

Definition 2.3 *An optimization problem $\Pi = (I, SOL, val, OPT)$ is in* **NPO** *if the following conditions are satisfied:*

- *there exists a polynomial p such that for all $x \in I, s \in SOL(x)$: $|s| \leq p(|x|)$,*
- *checking whether $s \in SOL(x)$ is in* **P***,*
- *computing $val(s)$ is in polynomial time.*

An **NP**-hard optimization problem is an **NPO** problem whose decision version is **NP**-complete. Max-2-Sat and TSP are well-known examples of **NP**-hard problem. An important subclass of **NPO**, denoted by **PO**, contains the **NPO** problems that are solvable in polynomial time.

Definition 2.4 **PO** *is the class of optimization problems Π such that:*

- $\Pi \in$ **NPO***,*
- *there is a polynomial-time algorithm that, for every instance x of Π, computes an optimal solution for x.*

In the literature, there are several approaches for dealing with **NP**-hard problems such as *exact algorithms, heuristics*, *approximation algorithms* and *randomized algorithms.* Each of these methods has its own advantages and could be efficiently applied for many **NP**-hard problems. Here we make a short overview of these approaches. More details can be found in the textbook of Hromkovič [Hro01].

Exact algorithms: The naive approach for solving optimization problems exactly is a brute-force search (a.k.a exhaustive search). Given an instance of an optimization problem, a brute-force algorithm will find all feasible solutions and check which one is optimal for that instance. This type of algorithm, of course, is easy to implement but the price to pay for this is the growing very quickly of the running time when the size of instance increases. A natural question is whether there is an algorithm significantly faster than brute-force one. In the detail, if the best exponential running time we can hope for exact algorithms is $\mathcal{O}(\alpha^n)$ for some $\alpha > 1$, then what is the best possible smallest value of α? Table 2.3 illustrates the improvement of exact exponential algorithms for some typical **NP**-hard problems. For more detail on the exact algorithms, we refer to the textbook of Fomin and Kratsch [FK10], the surveys of Woeginger [Woe01] and of Schöning [Sch05].

Table 2.3: Exact exponential algorithm for some **NP**-hard problems

Problem	Time	Reference	Time	Reference
3-SAT	$\mathcal{O}(1.6181^n)$	[MS85]	$\mathcal{O}(1.465^n)$	[Sch08]
INDEPENDENT SET	$\mathcal{O}(1.2599^n)$	[TT77]	$\mathcal{O}(1.1893^n)$	[Rob01]
3-COLORING	$\mathcal{O}(1.4422^n)$	[Law76]	$\mathcal{O}(1.3289^n)$	[BE05]
3-DOMATIC NUMBER	$\mathcal{O}(2.8805^n)$	[BH06]	$\mathcal{O}(2.695^n)$	[RRSY07]

Heuristics: The heuristic method was first used in the early 1940s by Polya [Pol48] and has been studied extensively in the literature. Unlike exact algorithms, the heuristic methods do not aim at computing optimal solutions, but return near-optimal ones. Usually, these solutions can be computed fast and easily but there no guarantee for their quality. For example, we consider the greedy-type heuristics for the problem TSP[2]. The greedy rule is that whenever we are at the vertex i, we will choose the next vertex j that is not chosen yet and has the smallest distance to i. An overview of heuristics method can be found in [RP85] and [Sil04].

[2]Note that the best known exact algorithm so far for TSP has running time of $\mathcal{O}(n^2 2^n)$, based on the dynamic programming.

Approximation algorithms: Basically, approximation algorithms are heuristic methods which produce approximation solutions with a bound on the quality of them. In the other words, if we can give a bound on the solution returned by a heuristic method, then we obtain an approximation algorithm. In practice, however, this is not always done. For example, it is known that TSP cannot be polynomially approximated to any factor, unless $\mathbf{P} = \mathbf{NP}$, although there exist heuristic algorithms for it. Since the approximation algorithm is one of the main concepts in this thesis, we devote the section 2.3 to present formal definitions and important properties.

Randomized algorithms: A randomized algorithm is an algorithm whose steps are not deterministic. While the deterministic algorithms look at the input to decide what to do next, the randomized algorithms flip coins for every nondeterministic choice. Randomized algorithms are used in many cases where one cannot find an exact solution in a deterministic way in a reasonable time. They may be faster and easier to describe than deterministic algorithms. Moreover, in many situations, one can obtain a deterministic algorithm for a problem by converting a known randomized algorithm using standard derandomization techniques. For more details on randomized algorithms, we recommend to read the textbooks [MR95], [MU05] and [Hro05].

2.3 Approximation Algorithms

As mentioned earlier, many optimization problems of theoretical and practical interests are *intractable*, meaning the exact solutions for these problems is unlikely to determine in polynomial time in size of problem input. Improving exact algorithms might be a good choice, but it remains difficult in practice to solve the problem instances whose input size are large. Another potential approach is the use of heuristic methods. Although the experiments show that the heuristics often find good solutions within polynomial time, but in general, there is no proof for the quality of solutions they found. This weakness could be overcome by using approximation algorithms, which always derive a bound on the quality of the solutions they generate. In the other words, it allows us to see that how well the heuristics can be performed on all instances. The field of approximation algorithms has been studied intensively in theoretical computer science in the last few decades and is considered as one of the promising approaches to tackle hard problems. This section is devoted to give an overview of this field. We start with a formal definition of approximation algorithms for maximization and minimization problems.

Definition 2.5 *Let* Π *be a maximization problem (minimization problem) and a function* $\alpha : \mathbb{N} \to \mathbb{R}^+$ *with* $\alpha < 1$ *(*$\alpha > 1$*). An* α-approximation algorithm A *for* Π *is an algorithm such that for each instance* x *of* Π*,* A *runs in polynomial time in* $|x|$ *and produces a feasible solution* $A(x)$ *for* x *such that*

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

We call α the *approximation factor* of the algorithm A. Note that in literature α is also called with other names such as *approximation ratio, performance ratio, guarantee ratio* or *factor*. The factor α depends on the size of instance, such as $\log n$ (e.g., Minimum Set Cover [Joh74], etc.) or n^c for some $c > 0$ (e.g, Max Clique [BH92]). In other cases, the factor α might be a constant function such as $1 - \varepsilon$ for some $\varepsilon > 0$ (e.g, Max-k-Sat [Joh74]), and for these cases we say A is a constant approximation algorithm. We will denote by **APX** the class of problems which admit a constant approximation algorithm.

In fact, there exist problems that are extremely well approximated in the sense that, one can obtain an approximation algorithm within factors arbitrarily close to 1 for them. We call such algorithms as approximation schemes.

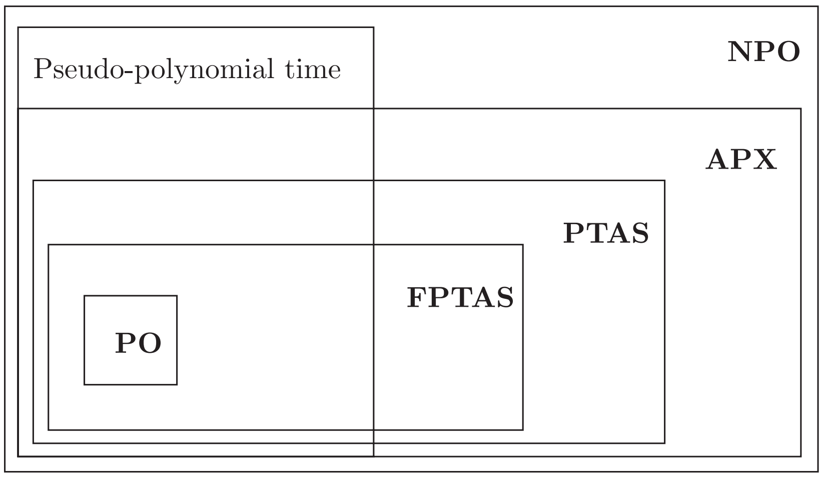
Definition 2.6 *A maximization problem (minimization problem)* Π *is said to have a* polynomial-time approximation scheme *(***PTAS***) if there exists an algorithm* A *such that for each* ε*,* $0 < \varepsilon < 1$*,* A *is* $(1 - \varepsilon)$*-approximation algorithm (*$(1 + \varepsilon)$*-approximation algorithm) for* Π.

Definition 2.7 *A maximization problem (minimization problem)* Π *has a* fully polynomial-time approximation scheme *(***FPTAS***) if for each* ε*,* $0 < \varepsilon < 1$*, there exists a* $(1 - \varepsilon)$*-approximation algorithm (*$(1 + \varepsilon)$*-approximation algorithm)* A_ε *for* Π*, where the running time is polynomial in* $1/\varepsilon$ *as well.*

The crucial difference between **PTAS** and **FPTAS** is the requirement of the running time bound on $1/\varepsilon$. For instance, if for each input of size n the algorithm A_ε runs in time $\mathcal{O}(n^{1/\varepsilon})$ then we have a **PTAS**, not an **FPTAS**. On the other hand, if the running time is $\mathcal{O}(n^{10}\varepsilon^{-100})$ then we have an **FPTAS**. Assuming $\mathbf{P} \neq \mathbf{NP}$, an **FPTAS** (a **PTAS**) is the best result we can obtain for a **NP**-hard problem (a strongly **NP**-hard problem[3]).

[3]An **NPO** problem is strongly **NP**-hard if it is **NP**-hard even if the input instance is specified in unary representation.

Figure 2.2: A classification of optimization problems in **NPO** according to their approximability.



From the computational point of view, we can precisely classify **NPO** problems based on their approximability.

$$\mathbf{PO} \subseteq \mathbf{FPTAS} \subseteq \mathbf{PTAS} \subseteq \mathbf{APX} \subseteq \mathbf{NPO}$$

We emphasize that, under the assumption of $\mathbf{P} \neq \mathbf{NP}$, all these inclusions are strict. Indeed, the problem Knapsack[4] is known to be in **FPTAS** (see [IK75]) but not in **PO** due to the fact that the underlying decision problem was shown to be **NP**-complete (see [Kar72]). Another problem, that belongs to **PTAS** but does not belong to **FPTAS**, is Euclidean TSP (see [Aro98]). It is considered as a variant of TSP where each node of graph is presented as a point in the Euclidean space and the distance between the two nodes is the Euclidean distance between the corresponding points. A representative example of the class **APX** is Max-k-Sat ($k \geq 1$). It was shown in [Joh74] that this problem is in **APX**, and it was shown in [PY91] that it does not belong to **PTAS**. Besides the classes **APX**, **PTAS**, **FPTAS**, there is a class of problems that are *pseudo*-polynomial time solvable. Intuitively, an algorithm for a problem Π is called pseudo-polynomial time if its running time is always bounded by a polynomial in size of problem instance and in size of largest element in the instance. Figure 2.2 illustrates the relationship between the complexity classes of **NPO** problems.

[4]Given a finite set A, a size $s(a) \in \mathbb{N}^+$ and a value $v(a) \in \mathbb{N}^+$ for every $a \in A$, and a positive integer B, find a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) \leq B$ and such that $\sum_{a \in A'} v(a)$ is maximized.

Techniques for designing approximation algorithms

Many different techniques are used to design approximation algorithms, which can be broadly categorized into the following classes: combinatorial algorithms, linear programming, semi-definite programming, randomized (and derandomized) algorithms. Within each class, standard algorithmic designs techniques such as local search, divide and conquer, greedy method, and dynamic programming are often adopted with a great deal of ingenuity. In literature, one can find many useful textbooks that cover this topic, and we recommend to, among others, the ones of Vazirani [Vaz03], Hochbaum [Hoc95], Ausiello et al. [GA99] and Williamson and Shmoys [DW11]. In the following we briefly sketch two standard techniques that will be used for designing approximation algorithms in this thesis, namely, greedy algorithm and dynamic programming.

Greedy: This algorithm is used for constructing successively a set of solutions, based on making locally optimal decision at each step of the algorithm, towards a global optimum. The advantages of using greedy algorithm lie in the fact that it is very easy to design and implement and especially requires much less amount of computational resources in practice. In some cases, the greedy method yields optimal solutions (e.g, Minimum Spanning Tree, using Prim's algorithm or Kruskal's algorithm). In some other cases, it returns good approximation solutions. For instance, consider the problem Load Balancing where we wish to assign a set of m jobs to n identical machines in such a way that the time in which all jobs are finished is as small as possible, given that executing a job j on any of n machines takes time p_j ($p_j > 0$ for all j). A greedy algorithm for the problem is followed by the rules: for each job j, $1 \leq j \leq m$, we assign it to the machine whose current load is smallest. Obviously, the running time of the algorithm is polynomial time in m and n. This algorithm is known as *List Scheduling* algorithm which was introduced by Graham [Gra66, Gra69]. Furthermore, Graham proved that this greedy algorithm gives a 2-approximation. To the best of our knowledge, List Scheduling is the first polynomial-time approximation algorithm known for a **NP**-hard problem. Another simple example of greedy approximation algorithm is for a variant of Traveling Salesman Problem, say Metric TSP, where the distances between the nodes of the graph satisfy the triangle inequality, see [CLRS09]. The algorithm is based on the solving exactly a Minimum Spanning Tree and has an approximation factor of 2. Many other approximation algorithms using the greedy technique can be found in the textbooks of Ausielo et al. [ACG$^+$99] and of Williamson and Shmoys [DW11]. In this thesis, by greedy method, we will provide a first approximation algorithm for the problem of maximizing Nash product social welfare (see Theorem 3.11 and Theorem 3.12), and a PTAS for the case with two agents having the same additive utility functions (see Theorem 3.5).

Dynamic programming: This is a very useful technique for solving efficiently many optimization problems in which one can build up from locally optimal solutions of subproblems to obtain a globally optimal solution. The crucial point to use this technique is formulating the solution process as a recursion. The use of dynamic programming in designing approximation algorithms has been intensively studied in the past decades. Particularly, it is an efficient tool to design (fully) polynomial-time approximation scheme for a variety of optimization problems. For example, by simplifying the input data in a suitable way (e.g, rounding a number x to the nearest integer or merging the two numbers into one of the same value, or even deleting a part of unnecessary data), we can transform an original instance into a more easy-to-tackle instance. Thereby, it can be solved by applying dynamic programming and then obtain a good approximation solution to the original instance. This approach was first used in the work of Horowitz and Sahni [HS74] to solve PARTITION problem. Sahni [Sah76] applied it to provide a PTAS for MINIMUM MAKESPAN on two machines. Also, based on this approach, Hochbaum and Shmoys [HS87] designed an approximation scheme for MINIMUM MAKESPAN on parallel machines and Arora [Aro98] proposed a PTAS for EUCLIDEAN TSP. In this thesis, by applying the same approach, we will give a PTAS for MAX-ANSW problem for the case of identical agents (see Theorem 3.7).

Another remarkable approach proposed by Ibarra and Kim [IK75] adds a procedure to the execution of the dynamic programming algorithm. This procedure will remove some unnecessary solutions of subproblems during the execution of the algorithm and allows us to keep the algorithm's memory as low as possible. As a consequence, the algorithm may return the inexact solution but it runs in polynomial time and we are able to obtain a good approximation. Approximation schemes for many scheduling problems by using this approach can be found in the sequence of papers [Sah76, PW92, KK98, KPW94]. In this thesis, we use again this approach to present an FPTAS for the problems MAX-ESW, MAX-TNSW, MAX-ANSW, and MINIMUM ENVY $(\mathrm{opt}_1, \mathrm{opt}_2)$.

Techniques for proving hardness of approximation

For many optimization problems, it is even hard to approximate them to a certain factor. Now, the question is, given an optimization problem, for which values of α does there not exist an α-approximation algorithm which runs in polynomial time? There are several useful techniques for dealing with such question. Among those, perhaps the most oldest and simplest one is using a so-called *gap-introducing reduction*. This type of reduction is much stronger than the polynomial-time many-one reduction that used to prove the **NP**-completeness of decision problems. It was used for the first time in [SG76] to prove the non-existence of any polynomial-time $f(n)$-approximation algorithm for the general TSP for any polynomial-time f.

Definition 2.8 (α-gap-introducing reduction) *Let A be a* **NP***-complete problem, Π be an optimization problem, and α 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 instance x of A, $g(x)$ is an instance of Π such that:*

1. *If Π is a maximization problem then $\alpha(|x|) < 1$ and:*
   - *if x is yes-instance of A then $OPT(g(x)) \geq f(x)$, and*
   - *if x is no-instance of A then $OPT(g(x)) < \alpha(|x|) \cdot f(x)$.*
2. *If Π is a minimization problem then $\alpha(|x|) > 1$ and:*
   - *if x is yes-instance of A then $OPT(g(x)) \leq f(x)$, and*
   - *if x is no-instance of A then $OPT(g(x)) > \alpha(|x|) \cdot f(x)$.*

Note that an α-approximation algorithm B for a maximization problem Π that has a α-gap-introducing reduction from a **NP**-complete problem A implies $x \in A$ if and only if the value of the solution $B(g(x))$ is greater than $\alpha(|x|) \cdot f(x)$. Hence, there can be no α-approximation algorithm for Π, unless **P** = **NP**. This important technique is also used to prove the inapproximability for most of our optimization problems in this thesis.

Another technique to give hardness results for **NPO** problems is via an *L-reduction*, that was introduced by Papadimitriou and Yannakakis [PY91], from an optimization problem that is known to be **NP**-hard to approximate within certain factors. This is the current most successful reduction for proving the hardness of approximation. The formal definition below is stated for maximization problems only, but would be similar to the case of minimization ones.

Definition 2.9 (L-reduction) *Let* Π_1 *and* Π_2 *be 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, $s_1 = g(s_2)$ *is a solution for* x *such that*

$$OPT(x) - val(s_1) \leq \beta \cdot (OPT(y) - val(s_2)).$$

Having an L-reduction from 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.

All hardness results presented in this thesis involve the use of two types of reductions above. In order to get the stronger inapproximability results, it might be useful to take other techniques into account such as: *Probabilistically Checkable Proofs* (PCPs) (see [AS98] and [ALM$^+$98]), using the reduction from *Label Cover* problem (see [ABSS93]) and using the *Unique Games Conjecture* that has been made recently by Subhash Khot [Kho02]. For more details, we refer to a textbook by Vazirani [Vaz03], the survey of Arora and Lund [AL96], the paper of Dinur [Din07] and the thesis of Arora [Aro94].

Chapter 3

Social Welfare Maximization Problems

Social welfare maximization problems can be described as a class of combinatorial optimization problems that seek to determine allocations of resources between agents, so as to maximizing social welfare. For that reason, we also call them resource allocation problems. Based on the type of social welfare being considered, a variety of resource allocation problems can be investigated. Among those, we focus on the following problems, each corresponds to a type of social welfare: Maximum Utilitarian Social Welfare, Maximum Egalitarian Social Welfare, Maximum Total Nash Social Welfare and Maximum Average Nash Social Welfare. The **NP**-completeness results for social welfare maximization problem under almost all naturally appearing type of social welfare and cardinal preferences motivate the study of approximation algorithms for these problems. The results in this chapter have been published in [NRR12, NNRR12b, NNRR12a, NR13b, NRR, NNRR].

3.1 The Framework and Basic Definitions

Let $A = \{a_1, a_2, \dots, a_n\}$ be a set of n agents and $R = \{r_1, r_2, \dots, r_m\}$ be a set of m indivisible and nonshareable resources. Subsets of R are called *bundles of resources*. Every agent associates utility to every bundle of resources by specifying a utility function $u_i : 2^R \to \mathbb{Q}^+$. We denote by $U = \{u_1, u_2, \dots, u_n\}$ the set of the agents' utility functions. A triple (A, R, U) is called a *multiagent resource allocation setting* (a *MARA setting*, for short). Given a MARA setting, we define an allocation of resources as follows:

Definition 3.1 *An* allocation *is a mapping* $\pi : A \to 2^R$, $a_i \mapsto \pi(a_i) = \pi_i$ *such that*

- $\bigcup_{i=1}^{n} \pi_i = R$ *(i.e., every resource is given to some agent)*

- *$\pi_i \cap \pi_j = \emptyset$ for $i \neq j$ (i.e., no resources are given to multiple agents).*

Equivalently, an allocation is a partition of the set of m resources R into n subsets. We denote an allocation by $\pi = (\pi_1, \dots, \pi_n)$, where π_i is the bundle assigned to agent a_i. If $\Pi_{A,R}$ is the set of all possible allocations for a MARA setting (A, R, U) with $\|A\| = n$, $\|R\| = m$ then $\|\Pi_{A,R}\| = n^m$.

3.1.1 Representations of Utility Functions

As mentioned in the previous chapter, utility functions can be represented in different forms, each has advantages and disadvantages depending upon the context of its use. Particularly, the representation form potentially affects the complexity of the corresponding allocation problems. In this thesis we will focus on the bundle form and the k-additive form which are defined formally as follows.

The bundle form: A utility function $u : 2^R \to \mathbb{Q}^+$ is in *bundle form* if it is represented by a list of pairs $(R', u(R'))$ for any bundle $R' \subseteq R$, omitting pairs with zero utility. This representation form is "fully expressive" (i.e., every utility function can be described in bundle form), but its drawback is a potentially exponential representation size in the number of resources.

The k-additive form (for some fixed positive integer k): A utility function $u : 2^R \to \mathbb{Q}^+$ is in *k-additive form* if for each bundle $T \subseteq R$ with $\|T\| \leq k$, there is a unique coefficient $\alpha^T \in \mathbb{Q}^+$ such that for every bundle $R' \subseteq R$ the following holds:

$$u(R') = \sum_{T \subseteq R', \|T\| \leq k} \alpha^T. \tag{3.1}$$

This coefficient α^T expresses the "synergetic" value of some agent owning all the resources in T. This representation form is fully expressive only if k is large enough. On the other hand, choosing k to be relatively small allows for a relatively succinct representation of utility functions. Originally, Grabisch [Gra97] defined the k-additive form. However, in multiagent resource allocation it was proposed for representing utilities by Chevaleyre et al. [CEEM04, CEEM08] and, independently, in combinatorial auctions by Conitzer et al. [CSS05].

The class of additive utility functions is a special case of the class of 1-additive utility functions, where the empty bundle has always zero utility for all agents.

Formally, a utility function u is additive if and only if for every bundle $R' \subseteq R$, we have:

$$u(R') = \sum_{r \in R'} u(r).$$

In practice, additive utilities are computationally much more appealing than general utility functions. Therefore, a lot of resource allocation problems in the literature have been studied under an assumption that agents' utility functions are additive.

Both the bundle and the k-additive form (for large enough k) can be used to represent any utility function and thus are equivalent in this sense. However, while the bundle form is a direct representation that enumerates the agents' nonzero utilities of bundles, the k-additive form can be seen as an indirect representation using the so-called *Möbius inversion* (see the work of Rota [Rot64] and Grabisch [Gra97]). Indeed, letting u_i be a set function over the set R of resources, the coefficients α_i^S for $S \subseteq R$ can be computed as follows:

$$\alpha_i^S = \sum_{T \subseteq S} (-1)^{\|S-T\|} \cdot u_i(S).$$

These two representations can be applied to any set function. The major difference between them is that the bundle form is just one fixed way of specifying utility functions, whereas the parameter k in the k-additive form allows one to fine-tune the trade-off between expressiveness (for large values of k) and compactness (for small values of k).

The following example demonstrates the representation of utilities in the bundle and the k-additive form.

Example 3.1 *Consider a MARA setting $M = (A, R, U)$ with $A = \{a_1, a_2\}$ and $R = \{r_1, r_2, r_3\}$ and the agents' utilities as stated in Table 3.1.*

In the bundle form, these utilities are represented as

$$(\{r_2\}, 3),\ (\{r_3\}, 2),\ (\{r_1, r_2\}, 3),\ (\{r_2, r_3\}, 5),\ (\{r_1, r_2, r_3\}, 3)$$

for agent a_1, and for agent a_2 as

$$(\{r_1\}, 2),\ (\{r_2\}, 1),\ (\{r_1, r_2\}, 3),\ (\{r_1, r_3\}, 2),\ (\{r_2, r_3\}, 1),\ (\{r_1, r_2, r_3\}, 3).$$

For the k-additive form, we adopt the notation from [CEEM04]. We first give the coefficient α from (3.1) and then give the single resources of the corresponding bundle.

Table 3.1: Utilities of the agents for Example 3.1

Bundles of Resources	Agent a_1	Agent a_2
$\emptyset$	0	0
$\{r_1\}$	0	2
$\{r_2\}$	3	1
$\{r_3\}$	2	0
$\{r_1, r_2\}$	3	3
$\{r_1, r_3\}$	0	2
$\{r_2, r_3\}$	5	1
$\{r_1, r_2, r_3\}$	3	3

While for the utilities of agent a_1 the 2-additive form is needed in our example, the utilities of agent a_2 can be given 1-additively. Omitting coefficients that are zero, we write $3.r_2 + 2.r_3 - 2.r_1.r_3$ to represent the utilities of agent a_1 and $2.r_1 + 1.r_2$ to represent those of agent a_2.

3.1.2 Measures of Social Welfare

The notion of social welfare is a tool to assess and rank allocations based on specific measures of quality. Thus, different allocations might be "the best allocation", depending on the notion of social welfare that is employed. The most fundamental notions of social welfare includes *utilitarian social welfare, egalitarian social welfare* and *total* and *average Nash (product) social welfare.* Assuming agents' preferences are specific by utility functions, we give the formal definition of these notions of social welfare.

Definition 3.2 *Let $\pi = (\pi_1, \ldots, \pi_n)$ be an allocation, we define*

1. *the* utilitarian social welfare of π *as* $sw_u(\pi) = \sum_{i=1}^{n} u_i(\pi_i)$.
2. *the* egalitarian social welfare of π *as* $sw_e(\pi) = \min_{1 \le i \le n} \{u_i(\pi_i)\}$.
3. *the* total Nash social welfare of π *as* $sw_N(\pi) = \prod_{i=1}^{n} u_i(\pi_i)$.
4. *the* average Nash social welfare of π *as* $sw_{\overline{N}}(\pi) = \left(\prod_{i=1}^{n} u_i(\pi_i)\right)^{\frac{1}{n}}$.

In this thesis, whenever we write "Nash social welfare", we mean any one of total and average version. As an additional notation, given a problem instance $M = (A, R, U)$, we denote the *maximum utilitarian/egalitarian/total Nash/average Nash social welfare of M* by

$$\max_t(M) = \max\{sw_t(\pi) \mid \pi \in \Pi_{A,R}\}.$$

for $t \in \{u, e, N, \overline{N}\}$.

Example 3.2 *Consider a MARA setting $M = (A, R, U)$ with $A = \{a_1, a_2, a_3\}$ and $R = \{r_1, r_2, r_3\}$, and agents' utility functions has 1-additive form which are given in Table 3.2.*

Table 3.2: Utilities of the agents for Example 3.2.

Single resources	Agent a_1	Agent a_2	Agent a_3
$\emptyset$	0	0	0
r_1	2	0	5
r_2	1	2	3
r_3	0	3	4

An allocation that maximizes utilitarian social welfare is that all resources are assigned to agent a_3 only, while the others get nothing. But if we want to optimize egalitarian social welfare, there are two alternatives: the first one is that the single resources r_1, r_2, r_3 are allocated to agents a_1, a_2, a_3, respectively, and the second one is that $r_1 \to a_1, r_3 \to a_2, r_2 \to a_3$. Note that the latter allocation is maximizing Nash social welfare.

3.1.3 Problem Definitions

After introducing the notions of social welfare and the type of representation of utility functions, the bundle form and the k-additive form, through which the agents' preferences are expressed, we now give the formal definition of social welfare maximization problems that are considered in this chapter. The first problem, denoted by MAX-USW, was studied originally in the context of combinatorial auction where the (utilitarian) social welfare has to be maximized. Formally, the problem is defined as follows:

Maximum Utilitarian Social Welfare (Max-USW)	
Input:	A MARA setting $M = (A, R, U)$ where A is a set of n agents, R is a set of m resources and U is a set of agents' utility functions.
Output:	An allocation $\pi = (\pi_1, \dots, \pi_n)$ such that $\sum_{i=1}^{n} u_i(\pi_i)$ is maximal.

In contrast to Max-USW, the problem of maximizing egalitarian social welfare, which is denoted by Max-ESW, concerns the question of how to assign resources to agents so that the minimum of the sum of the values of the resources given to any agent is maximal. This problem was indeed first studied as a machine scheduling problem where the goal is to find an allocation of jobs between machines such that the minimum completion time is maximized (see [Woe97]).

Maximum Egalitarian Social Welfare (Max-ESW)	
Input:	A MARA setting $M = (A, R, U)$ where A is a set n of agents, R is a set of m resources and U is a set of agents' utility functions.
Output:	An allocation $\pi = (\pi_1, \dots, \pi_n)$ such that $\min_{i=1}^{n} u_i(\pi_i)$ is maximal.

The next two problems, denoted by Max-TNSW and Max-ANSW, aim to maximize the total and average Nash social welfare of the society, respectively. Interestingly, while the first two problems Max-USW and Max-ESW has been studied deeply for a long time, the approximability of Max-TNSW and Max-ANSW have not been investigated so far.

Maximum Total Nash Social Welfare (Max-TNSW)	
Input:	A MARA setting $M = (A, R, U)$ where A is a set of n agents, R is a set of m resources and U is a set of agents' utility functions.
Output:	An allocation $\pi = (\pi_1, \dots, \pi_n)$ such that $\prod_{i=1}^{n} u_i(\pi_i)$ is maximal.

Maximum Average Nash Social Welfare (Max-ANSW)	
Input:	A MARA setting $M = (A, R, U)$ where A is a set of n agents, R is a set of m resources and U is a set of agents' utility functions.
Output:	An allocation $\pi = (\pi_1, \dots, \pi_n)$ such that $(\prod_{i=1}^{n} u_i(\pi_i))^{1/n}$ is maximal.

It is worth pointing out that, although the two problems Max-TNSW and Max-ANSW are obviously the same in the sense of computing exact optimal solutions, it is not clear that whether their (in)approximability are transferred mutually. For instance, despite the fact that Max-ANSW with the additive form admits a PTAS for the case of identical agents (see Theorem 3.7), it is still not known whether or not the same result can be applied to Max-TNSW.

3.2 State of the Art

The approximability and inapproximability of social welfare maximization problems have been studied extensively in the past few decades, but most particularly during the last few years. This section is devoted to give a short summary of the known results in this area from literature. The results will be presented separately for each type of representation of utility functions, either the bundle form or the k-additive form.

3.2.1 The Bundle Form

In the bundle form, the problem of maximizing utilitarian social welfare was studied deeply in the context of combinatorial auctions where a set of indivisible items are sold to bidders and each bidder may place bids on arbitrary subsets of those items. The goal of the auctioneer is to partition the items amongst bidders in a way that maximizes the revenue. In general, computing such a partition is **NP**-hard (see Rothkopf et al. [RPH98] and Dunne et al. [DWL05]). Furthermore, Lehmann, O'Callaghan, and Shoham [LOS99] proved that it cannot be approximated in polynomial time within a factor of $m^{-1/2+\varepsilon}$ unless $\mathbf{NP} = \mathbf{ZPP}$[1], where m is the number of items. The lower bound holds even for the case of *single-minded bidders* via an L-reduction from Maximum Clique which is hard to approximate (see Håstad [Hås99]). Formally, given a set R of m items, a bidder a_i is said to be *single-minded* if there is a bundle $R_i \subseteq R$ and a value $v \in \mathbb{Q}^+$ such that $u_i(T) = u_i(R_i) = v$ for any bundle T with $R_i \subseteq T \subseteq R$, and $u_i(T) = 0$ otherwise. Therefore, a combinatorial auction with n single-minded bidders and a set R of items can be expressed by a sequence $(R, (R_1, u_1), \ldots, (R_n, u_n))$, where u_i is the value of the "core bundle" R_i to bidder a_i. Independently, Roos and Rothe [RR10] provided a similar inapproximability result for Max-USW, but in the context of resource allocation, following from a result of Chevaleyre et al. [CEEM08].

[1]**ZPP** has been introduced by Gill [Gil77] as the class of decision problems that have a probabilistic algorithm that, on every input, runs in expected polynomial time and never returns a wrong answer.

Due to the hardness of MAX-USW, many attempts have been made to find special cases in which the problem can be solved or approximated efficiently. For example, instead of considering general utility functions, we can restrict those to *submodular* functions or *subadditive* functions[2]. The main reason for using these functions comes from the properties themselves. Namely, they have similarly properties to convex functions in optimization theory, and their structural characteristics can be taken advantage of algorithmically. In other words, submodularity or subadditivity of functions can help to make the related optimization problems approximable or even tractable.

In general, utility functions represented in the bundle form might be exponential in the number of resources. Therefore, we need to specify how an algorithm can access its input. Typically, we can resort to an *oracle model* that can answer a certain type of queries about a utility function. The most simple model of oracle is *value oracle*, which returns the utility of a given bundle of resources for a given agent.[3] In this model, Lehmann, Lehmann, and Nisan [LLN01] designed a greedy algorithm that achieves an approximation factor of $1/2$ for MAX-USW, assuming utility functions are submodular. The key idea of their algorithm is that every resource will be assigned to an agent with maximal marginal value in the current allocation. Fleischer et al. [FGMS06], Calinescu et al. [CCPV07], and Vondrák [Von08] improved their result to a $(1-1/e)$-approximation, where $e \approx 2,71828182$ is Euler's number. Khot et al. [KLMM08] proved that this bound of $1 - 1/e$ is tight: It is **NP**-hard to approximate MAX-USW with a factor better than $1 - 1/e$ in the submodular setting. For the class of subadditive functions, Dobzinski, Nisan, and Schapira [DNS10] presented a $(1/\sqrt{m})$-approximation algorithm for MAX-USW. Our survey [NRR] provides a detailed summarization of the results that have been obtained so far for MAX-USW (most of them are in the context of combinatorial auctions).

The approximability of MAX-ESW with respect to the bundle form was studied by the authors Golovin [Gol05], Goemans et al. [GHIM09] and Khot and Ponnuswami [KP07] in the model of value oracle. Golovin [Gol05] gave a greedy approximation algorithm for this problem with a factor of $1/(m-n+1)$ based on a matching technique, when utility functions are submodular. This approximation factor was then improved to $1/\left(m^{1/2}n^{1/4}\log m \log^{3/2} n\right)$ by Goemans et al. [GHIM09]. The idea is to transform the original problem to the *Santa Claus problem*, which was shown to be approximable by Asadpour and Saberi [AS07, AS10]. In particular, when the number of agents is fixed, Chekuri,

[2]see the book of Schrijver [Sch03] for formal definitions.

[3]Other types of oracle such as demand oracle or general oracle was also studied in the context of combinatorial auction.

Table 3.3: Summary of known (in)approximability results for the social welfare optimization problems for the bundle form, using the model of value oracle.

Problem	Submodular	Reference	Subadditive	Reference
MAX-USW	$1/2$	[LLN01]	$1/\sqrt{m}$	[DNS10]
	$1 - 1/e$ (randomize)	[Von08]		
MAX-ESW	$1 - 1/e - \varepsilon$ ($n = const$)	[CVZ10]	$1/(2n-1)$	[CVZ10]
	$1/(m-n+1)$	[Gol05]		
	$1/(m^{1/2} n^{1/4} \log m \log^{3/2} n)$	[GHIM09]		

Vondrák, and Zenklusen [CVZ10] gave an approximation algorithm for MAX-ESW to within a factor of $1 - (1/e) - \varepsilon$, for any $\varepsilon > 0$. Khot and Ponnuswami [KP07] studied the case of subadditive utility functions and they propose a $1/(2n-1)$-approximation algorithm.

3.2.2 The Additive Form

In terms of additive form, the problem MAX-USW is solvable in polynomial time. A simple algorithm for that follows by the rule: every resource will be assigned to an agent who has maximum utility for it. Unlike MAX-USW, however, MAX-ESW is **NP**-hard even for the special case of identical agents, i.e, the value of each resource does not depend on the agent to which it is assigned. The best inapproximability result known so far for MAX-ESW is due to Bezáková and Dani [BD05]: it cannot be approximated in polynomial time within a factor of $\alpha > 1/2$, unless $\mathbf{P} = \mathbf{NP}$. In the same paper, using techniques based on matching as well as rounding techniques for linear programming relaxation, Bezáková and Dani designed two approximation algorithms for MAX-ESW, both having a performance guarantee of $1/(m-n+1)$. Moreover, using linear programming techniques, introduced by Lenstra et al. [LST90], the authors proved that one can obtain a solution of value at least $OPT - \max_{ij} u_i(r_j)$, where OPT denotes the value of the optimal solution. However, the challenging case of the problem is when the condition $OPT \leq \max_{ij} u_i(r_j)$ holds.

Golovin [Gol05] studied a restricted version of MAX-ESW, which is also known as "*Big Goods/Small Goods*." In this restricted problem, the agents are allowed to choose among three values only (0, 1, and some $x > 1$) to express their utilities. For the *small* goods, each agent is allowed to assign utilities of zero and one, and for the *big* goods the allowed

Table 3.4: Summary of known approximability results for the problem of maximizing egalitarian social welfare for the additive form.

MAX-ESW	Approximability	Reference
general	$1/(m-n+1)$	[BD05]
	$1/n$	[Gol05]
	$\Omega(1/\sqrt{n}\log^3 n)$	[AS07, AS10]
	$1/m^{\varepsilon}$ (for any $\varepsilon = \Omega(\log\log m/\log m)$)	[CCK09]
identical agents	PTAS	[Woe97]
model of *Big Goods/Small Goods*	$1/\sqrt{n}$	[Gol05]
$m = n$	**P**	[Gol05]
$u_i(r_j) \in \{0,1\}$ for all i,j	**P**	[Gol05]
$u_i(r_j) \in \{0,x_j\}$ for all i,j	$\mathcal{O}(\log\log\log n/\log\log n)$	[BS06]
$u_i(r_j) \in \{0,1,x\}$ for all i,j	α/n (for any $\alpha \leq n/2$)	[KP07]

values are zero and $x > 1$. Using min-cut and network flow techniques, Golovin [Gol05] showed that this problem is approximable in polynomial time within a factor of $1/\sqrt{n}$. Khot and Ponnuswami [KP07] generalized this model of *Big Goods/Small Goods* by considering the less restricted version in which the agents' utilities for a resource are either 0, 1, or x for some $x > 1$. For this case, they gave an (α/n)-approximation algorithm that runs in time $m^{\mathcal{O}(1)}n^{\mathcal{O}(\alpha)}$. Golovin [Gol05] also studied another special variant of MAX-ESW where there are as many agents as resources (i.e., $m = n$). It is then easy to see that each agent must get at least one resource to obtain an egalitarian social welfare distinct from zero. Hence, this problem can be solved in polynomial time [Gol05].

Bansal and Sviridenko [BS06] investigated another restriction of MAX-ESW under the name of the Santa Claus problem: If only two values are allowed for each single resource (i.e., r_j has either some value x_j or zero for each of the agents), then there is an $\mathcal{O}(\log\log\log n/\log\log n)$-approximation. Their method was based on rounding a certain type of linear programming (LP) relaxation that was also known as *configuration LP*. Later on, Feige [Fei08] showed that the value obtained by this LP relaxation estimates the optimum value to within a constant (unspecified) factor. However, his algorithm is not constructive, that is, it does not allow to actually find the corresponding allocation in polynomial time. At the same time, Asadpour, Feige and Saberi [AFS08] proved that the integrality gap of the configuration LP is no worse than $1/5$ (this factor was then improved to $1/4$ in [AFS12], the journal version of [AFS08]). Their approach was based on the local search algorithms for computing perfect matchings in certain classes of hypergraphs. Nevertheless, the local search algorithm they provided for finding a ($1/4$)-approximation allocation takes 2^n steps and thus, is not a polynomial-time algorithm.

In the general case, Asadpour and Saberi [AS07, AS10] proposed an improved approximation algorithm that achieves a factor of $\Omega(1/\sqrt{n}\log^3 n)$ for MAX-ESW, using the same LP relaxation that Bansal and Sviridenko [BS06] employed. The best known approximability result so far for the general problem was made by Chakrabarty et al. [CCK09]: They presented an $\mathcal{O}(1/m^{\varepsilon})$-approximation algorithm for any $\varepsilon \in \Omega(\log\log m/\log m)$ that runs in time $m^{\mathcal{O}(1/\varepsilon)}$. Note that this result can also be used to obtain an approximation in terms of the number of agents.

Building and improving on the work of Deuermeyer et al. [DFL82] and Csirik et al. [CKW92], Woeginger [Woe97] studied the problem MAX-ESW for the case where agents have the same utility on each single resource, and he designed a PTAS for it. Since this problem is **NP**-complete in the strong sense [GJ79], there can be no FPTAS for it, unless $\mathbf{P} = \mathbf{NP}$.

The known approximability results for the problem MAX-ESW with the additive form are summarized in Table 3.4. For the more detail, we refer to the survey [NRR].

3.3 Our Results

In this section, we analyse the approximability and inapproximability of the social welfare maximization problems. The results are presented according to whether they are negative or positive.

In section 3.3.1 we derive the lower bounds on the approximation of the problems MAX-USW, MAX-ESW, MAX-TNSW and MAX-ANSW, focusing on both types of representation of utility functions. We show that unless $\mathbf{P} = \mathbf{NP}$, MAX-USW and MAX-TNSW with 2-additive utilities are not in **PTAS**. In fact, there does not exist $(21/22 + \varepsilon)$-approximation algorithm for these two problems for any $\varepsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$. The slightly stronger lower bounds for MAX-TNSW and MAX-ANSW with additive utilities are obtained via an L-reduction. In particular, we prove that there is no approximation algorithm for MAX-ESW, MAX-TNSW and MAX-ANSW when utility functions are represented in the bundle form or in the 3-additive form.

In section 3.3.2 we provide the upper bounds for Max-ESW, Max-TNSW and Max-ANSW assuming additive utility functions. First we design a PTAS for Max-TNSW when there are only two identical agents and then extend this result to the case with any number of agents. The proof is based on integer programming, and relies on rounding the resources to the "big" and "small" ones. Second, we prove that our optimization problems have an FPTAS if the number of agents is constant. Finally, we propose a first approximation algorithm for Max-TNSW and Max-ANSW and present a polynomial-time algorithm for the case when the number of agents and the number of resources are the same.

3.3.1 Inapproximability Results

The bundle form

Roos and Rothe [RR10] raised the question of whether Max-ESW and Max-TNSW is as hard to approximate as Max-USW given the bundle form of agents' preferences. It turns out that the reduction from 3-Sat used to prove the **NP**-completeness of the decision versions of Max-ESW and Max-TNSW in [RR10] is sufficient to show the inapproximability of these optimization problems. In the proof of the proposition below we are using the reduction from another **NP**-complete problem, Exact Set Cover (XSC, for short), that may seem to be much more simple than the one of Roos and Rothe [RR10]. The results show that Max-ESW, Max-TNSW and Max-ANSW cannot be approximated in polynomial time within any factor, unless $\mathbf{P} = \mathbf{NP}$, whenever the preferences of agents are modelled in terms of the bundle form.

Exact Set Cover (XSC)	
Given:	A finite set $\mathcal{B} = \{b_1, \ldots, b_m\}$ and a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of subsets of $\mathcal{B}$.
Question:	Is there a subset $I \subseteq \{1, \ldots, n\}$ such that $\{S_i \mid i \in I\}$ is a partition of $\mathcal{B}$, i.e., $\bigcup_{i \in I} S_i = \mathcal{B}$ and $S_i \cap S_j = \emptyset$ for all distinct $i, j \in I$?

Theorem 3.1 *Unless* $\mathbf{P} = \mathbf{NP}$*, there is no* α*-approximation algorithm for the problems* Max-ESW*,* Max-TNSW *and* Max-ANSW *with the bundle form, for any factor* α*. This even holds when the utilities are restricted to the domain* $\{0, 1\}$*.*

Proof. It suffices to prove the claim for Max-ESW as the cases of Max-TNSW and Max-ANSW then follow directly. Let $(\mathcal{B}, \mathcal{S})$ with $\mathcal{B} = \{b_1, \ldots, b_m\}$ and $\mathcal{S} =$

$\{S_1, \ldots, S_n\}$ be an instance of XSC. Construct a new instance $M = (A, R, U)$ of Max-ESW as follows. Let $A = \{a_1, \ldots, a_n\}$ be our set of agents, and let $R = \mathcal{B}$ be our set of m resources. For each i, $1 \leq i \leq n$, the utility function of agent a_i is defined as

$$u_i(T) = \begin{cases} 1 & \text{if } T = \emptyset \text{ or } T = S_j \text{ for some } j \\ 0 & \text{otherwise.} \end{cases}$$

If $(\mathcal{B}, \mathcal{S})$ is a yes-instance of XSC then there exists some set $I \subseteq \{1, \ldots, n\}$ such that $\{S_i \mid i \in I\}$ is a partition of $\mathcal{B}$, i.e., $\bigcup_{i \in I} S_i = \mathcal{B}$ and $S_i \cap S_j = \emptyset$ for all distinct $i, j \in I$. Hence, by assigning the bundle S_i to agent a_i for each $i \in I$ and the empty bundle to all remaining agents, we have obtained an allocation which has the egalitarian social welfare of 1. It follows that $\max_e(M) \geq 1$.

Conversely, suppose $(\mathcal{B}, \mathcal{S})$ is a no-instance of XSC. As all resources need to be allocated, there is always at least one agent a_i, for some i, who is not assigned her preferred bundle S_i. This implies that the optimal social welfare value is zero: $\max_e(M) = 0$.

To sum up, if one could approximate in polynomial time to any factor for Max-ESW, one could thus decide XSC in polynomial time, contradicting **NP**-hardness of XSC unless **P** = **NP**. ❑

The k-additive form

In the k-additive form, Max-USW is in **P** if $k = 1$, but is **NP**-hard for any fixed $k \geq 2$. This result was given by Chevaleyre et al. [CEEM08], using a reduction from the decision version of Max-2-Sat. We will show that, for the 2-additive utilities, this is also an L-reduction from Max-2-Sat to Max-USW which immediately implies an inapproximability result for the latter problem .

Maximum 2-Satisfiability (Max-2-Sat)	
Input:	A boolean formula φ in conjunctive normal form consisting of clauses having two literals each.
Output:	A truth assignment to the variables of φ that maximizes the number of satisfied clauses.

The best (unconditional[4]) inapproximability result currently known for MAX-2-SAT is due to Håstad [Hås01], who shows that this problem is **NP**-hard to approximate within a factor of $21/22 \approx 0.9545$.

Proposition 3.1 MAX-USW *with the k-additive form, for a fixed $k \geq 2$, cannot be approximated in polynomial time to within any factor better than $21/22$, unless $\mathbf{P} = \mathbf{NP}$. This result holds even when there are only two agents.*

Proof. It suffices to prove the claim for $k = 2$ since the 2-additive form is a special case of the k-additive form for any $k > 2$.

Let φ be an instance of MAX-2-SAT. The MAX-USW instance $M = (A, R, U)$ constructed from φ has two agents (i.e., $A = \{a_1, a_2\}$), each resource in R corresponds to a propositional variable occurring in φ, and the agents' utilities are set in 2-additive form. The utility function of agent a_1 is defined by Chevaleyre et al. [CEEM08] as the sum of the following 2-additive terms each of which corresponds to a clause of φ:

- x_i if the clause has the form $x_i \vee x_i$,
- $(1 - x_i)$ if the clause has the form $\neg x_i \vee \neg x_i$,
- $x_i + x_j - x_i \cdot x_j$ if the clause has the form $x_i \vee x_j$, $i \neq j$,
- $x_i + (1 - x_j) - x_i(1 - x_j)$ if the clause has the form $x_i \vee \neg x_j$, $i \neq j$, and
- $(1 - x_i) + (1 - x_j) - (1 - x_i)(1 - x_j)$ if the clause has the form $\neg x_i \vee \neg x_j$, $i \neq j$.

Agent a_2's coefficients are defined as $\alpha_2^T = 0$ for all bundles $T \subseteq R$, $\|T\| \leq 2$.

Note that every assignment τ of truth values to the propositional variables of φ corresponds to a resource allocation X_τ for M, since agent a_1 receives resource x_i exactly if x_i is set to true under τ. It follows that $sw_u(X_\tau)$ equals the number of clauses in φ satisfied by τ. Thus $\max_u(M)$ equals the maximum number of satisfiable clauses in φ. Since this is an L-reduction from MAX-2-SAT to MAX-USW, the inapproximability bound of $21/22$ for MAX-2-SAT due to Håstad [Hås01] immediately transfers to MAX-USW. ❑

Unlike the problem of finding the maximum utilitarian social welfare, the problems of maximizing the egalitarian and Nash social welfare are **NP**-hard with respect to the additive form. This result is proved by using a gap-reduction from PARTITION

[4]Khot [KGMO07] show that MAX-2-SAT is **NP**-hard to approximate within a factor better than roughly 0.9439, provided that the so-called "Unique Games Conjecture" holds.

problem and holds even for very restricted case with only two agents having the same utility functions (see Bouveret et al. [BLFL05], Roos and Rothe [RR10], Lipton et al. [LMMS04]). As a consequence, Max-ESW, Max-TNSW and Max-ANSW are **NP**-hard for the k-additive form for any fixed $k \geq 1$.

The inapproximability of Max-ESW was studied first by Bezáková and Dani [BD05] nearly ten years ago. In fact, they showed a hardness factor of $1/2$ for this problem for additive utility functions. An improving of this bound is still a challenge now. For both Max-TNSW and Max-ANSW, there is no lower bound known so far. Note that the reduction from Partition for the **NP**-completeness proof does not yield any inapproximability results for Max-TNSW and Max-ANSW. In the following we will give the first lower bound results for these two optimization problems.

For the case of 2-additive utilities, one can get a first hardness result for Max-TNSW by modifying a bit the proof of Proposition 3.1. More detail, we change slightly the L-reduction from Max-2-Sat as follows: For agent a_2, the empty bundle has utility one and all other bundles with at most two resources have utility zero. Thus the number of clauses satisfied is exactly the maximum total Nash social welfare. Everything else remains unchanged. Using again the lower bound of $\alpha = 21/22$ for Max-2-Sat due to Håstad [Hås01], the result follows immediately.

We next propose a gap-reduction from the **NP**-complete problem Exact Cover by 3-Sets (or X3C, for short) which yields a first $8/9$-hardness result for Max-TNSW and another $2\sqrt{2}/3$-hardness result for Max-ANSW, when agents' utilities are additive. The formal definition of X3C is as follows:

Exact Cover by 3-Sets (X3C)	
Given:	A finite set $\mathcal{B}$ with $\|\mathcal{B}\| = 3q$ and a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of 3-element subsets of $\mathcal{B}$.
Question:	Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that every element of $\mathcal{B}$ occurs in exactly one member of $\mathcal{S}'$?

Theorem 3.2 Max-TNSW *with the additive form cannot be polynomially approximated within a factor better than* $8/9$, *unless* $\mathbf{P} = \mathbf{NP}$.

Proof. Let $(\mathcal{B}, \mathcal{S})$ with $\|\mathcal{B}\| = 3q$ and $\mathcal{S} = \{S_1, \ldots, S_n\}$ be an instance of X3C. Without loss of generality (w.l.o.g), assume that $n > q$. Construct an instance $M = (A, R, U)$ of Max-TNSW as follows. Let A be a set of n agents, where agent a_i corresponds to S_i, and let $R = \mathcal{B} \cup D$ be a set of $m = 2q + n$ resources. That is, there are $3q$ "real" resources that correspond to $3q$ elements of $\mathcal{B}$, and there are $n-q$ "dummy" resources in D. Define the agents' utilities as follows. For each $a_i \in A$ and each $r_j \in R$, define

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

Suppose that $(\mathcal{B}, \mathcal{S})$ is a yes-instance of X3C. Then there exists a set $I \subseteq \{1,\dots,n\}$, $\|I\| = q$, such that $S_i \cap S_j = \emptyset$ for all $i,j \in I$, $i \neq j$, and $\bigcup_{i\in I} S_i = \mathcal{B}$. Hence, we assign the bundle S_i to agent a_i for each $i \in I$, and the dummy resources to the $n-q$ remaining agents. This allocation maximizes the total Nash social welfare, which now is at least 1. Furthermore, the sum of all agents' individual utilities is at most n. 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 $(\mathcal{B}, \mathcal{S})$ is a no-instance of X3C, we show that the maximum total Nash social welfare is at most $8/9$. Note that the sum of all agents' utilities is still at most n in this case. Let $(\pi_1,\dots,\pi_n)$ be an optimal allocation for the instance M. As a shorthand (and slightly abusing notation), we denote by u_i the utility of agent a_i for his or her bundle π_i. Without loss of generality, we assume that $\sum_{i=1}^{n} u_i = n$. The goal is to find values $u_1,\dots,u_n$ that maximize the product $\prod_{i=1}^{n} u_i$, where $u_i \in \{1/3, 2/3, 1, 4/3, 5/3, 2\}$ for each i, $1 \leq i \leq n$. (Note that no agent a_i can receive more than one dummy resource, otherwise one could reassign the dummy resource to an agent who realizes less than utility one.)

If $u_i = 1/3$ for some i, then there exists a $j \neq i$ such that $u_j = 1 + (\ell/3)$ for some $\ell \in \{1,2,3\}$. Clearly, by replacing u_i and u_j in $\prod_{i=1}^{n} u_i$ by $u_i + (\ell/3)$ and $u_j - (\ell/3)$, respectively, we will get a product that is greater. By the same argument for the cases $u_i = 5/3$ and $u_i = 2$, we have that the value domain of the agents' utilities is reduced to $\{2/3, 1, 4/3\}$. Since $(\mathcal{B}, \mathcal{S})$ is a no-instance of X3C, there must be at least one agent whose bundle has a value less than 1. This implies that the product $\prod_{i=1}^{n} u_i$ has the form of $(8/9)^z$ for some integer $z \geq 1$ and reaches the maximal value of $8/9$ with $z = 1$. Therefore, an approximation algorithm with a factor better than $8/9$ will distinguish the "yes" and "no" instances of X3C. ❑

Theorem 3.3 MAX-ANSW *with the additive form cannot be polynomially approximated within a factor better than* $2\sqrt{2}/3$, *unless* $\mathbf{P} = \mathbf{NP}$.

Proof. We use again the gap-reduction as in the proof of Theorem 3.2. By the same argument, one can show that the value of an optimal solution for instance M constructed from the given instance $(\mathcal{B}, \mathcal{S})$ of X3C, equals 1 if $(\mathcal{B}, \mathcal{S})$ is yes-instance, and is less than or equal to $\sqrt[n]{8/9}$, otherwise. Note that $\sqrt[n]{8/9} \leq \sqrt{8/9} = 2\sqrt{2}/3$ for all $n \geq 2$. Hence, for any $\varepsilon > 0$, an $(2\sqrt{2}/3 + \varepsilon)$-approximation algorithm for Max-ANSW will decide X3C in polynomial time. This contradicts the **NP**-completeness of X3C. The theorem is proved. ❑

As the additive form is special case of the k-additive form for any $k \geq 1$, all the hardness results with respect to the former can be transformed immediately to the latter. Our next aim is to establish the stronger inappoximability results for the case with $k \geq 3$. In particular, given the agents' preferences described by the 3-additive utility functions, we show that the problems Max-ESW, Max-TNSW and Max-ANSW are inapproximable in polynomial time within any factor, unless $\mathbf{P} = \mathbf{NP}$. The proof is similar to the one of Theorem 3.1, but this time, the gap-reduction is from X3C instead of XSC.

Theorem 3.4 *If agents' utility functions are represented by the k-additive form for any fixed $k \geq 3$, there is no α-approximation algorithm for* Max-ESW*,* Max-TNSW *and* Max-ANSW *for any factor $\alpha < 1$, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. It suffices to prove the claim for the case of Max-ESW with the 3-additive from. Let $(\mathcal{B}, \mathcal{S})$ be an instance of X3C, where $\|\mathcal{B}\| = 3q$ and $\mathcal{S} = \{S_1, \dots, S_n\}$. Construct an instance $M = (A, R, U)$ of Max-ESW as follows. The set of q agents is $A = \{a_1, \dots, a_q\}$ and the set of resources is $R = \mathcal{B}$. For each agent $a_i \in A$, define the coefficients in the 3-additive representation of a_i's utility function as follows:

$$\alpha_i^T = \alpha^T = \begin{cases} 1 & \text{if } T = S_j \text{ for some } j \\ 0 & \text{otherwise.} \end{cases}$$

Suppose that $(\mathcal{B}, \mathcal{S})$ is a yes-instance of X3C. Then there exist n pairwise disjoint subsets $S_1, \dots, S_q$ of $\mathcal{S}$ such that $\bigcup_{1 \leq i \leq q} S_i = \mathcal{B}$. Hence, assigning the bundle S_i to agent a_i for each i, $1 \leq i \leq q$, we obtain an allocation that has the value of 1. Thus, $\max_e(M) \geq 1$.

Conversely, we show that if $(\mathcal{B}, \mathcal{S})$ is a no-instance of X3C, then there is at least one agent who owns a bundle $T \subseteq \mathcal{B}$ such that T does not contain any subset $S_i \in \mathcal{S}$. This implies that $sw_e(\pi) = 0$ for every allocation π, so $\max_e(M) = 0$. Indeed, assume that all agents are assigned bundles containing some $S_j \in \mathcal{S}$. Since the resources are indivisible and nonshareable, there must be n pairwise disjoint subsets in $\mathcal{S}$ that are an exact cover of $\mathcal{B}$, a contradiction.

Therefore, if there is a polynomial-time approximation algorithm that approximates Max-ESW within any factor, then it could distinguish yes- from no-instances of X3C. This contradicts the **NP**-hardness of X3C unless **P** = **NP**. ❑

3.3.2 Approximability Results

All approximation algorithms and exact polynomial-time algorithms in this section are achieved under an assumption that agents represent their preferences by the additive utility functions. However, some of the results can be transferred immediately to the case of the k-additive functions and the case of submodular functions. The techniques used for designing approximation algorithms include greedy algorithms, dynamic programming and integer programming.

A PTAS for Max-TNSW and Max-ANSW for the case with two identical agents

As mentioned earlier, even for the simplest case with two agents having identical utilities for the resources, Max-TNSW and Max-ANSW are still **NP**-hard. However, we can show that this case can be approximated efficiently to within any factor less than 1, by using greedy algorithm.

Theorem 3.5 Max-TNSW *admits a PTAS when restricted to only two agents having the same additive utility functions.*

Proof. Let $M = (A, R, U)$ be a problem instance with two agents (i.e., $A = \{a_1, a_2\}$) and nonnegative utilities $u_1(r) = u_2(r) \geq 0$ for all $r \in R$. Let u denote this utility function, i.e., $u = u_1 = u_2$. Consider the following greedy algorithm for our problem. Intuitively, this algorithm seeks to find disjoint subsets π_1 and π_2 of R such that $\pi_1 \cup \pi_2 = R$ and the product $u(\pi_1) \cdot u(\pi_2)$ is maximized, where $u(T) = \sum_{r \in T} u(r)$. Without loss of generality, we can assume that π_1 is assigned to agent a_1 and π_2 is assigned to agent a_2. Let ε be any fixed constant such that $0 < \varepsilon < 1$.

Algorithm 1 A greedy algorithm giving a PTAS for MAX-TNSW

```
1: Sort the resources in nonincreasing order of their utilities to get a sequence
   $(r_1, r_2, \dots, r_m)$ such that $u(r_1) \geq u(r_2) \geq \cdots \geq u(r_m)$.
2: Set $\ell := \lceil -1 + 1/\sqrt{1-\varepsilon} \rceil$.
3: Perform an exhaustive search for an optimal solution $(\pi_1, \pi_2)$ over the $\ell$ resources
   of $(r_1, r_2, \dots, r_\ell)$.
4: for $i := \ell + 1$ to $m$ do
5:   if $u(\pi_1) \leq u(\pi_2)$ then
6:     $\pi_1 := \pi_1 \cup \{r_i\}$
7:   else
8:     $\pi_2 := \pi_2 \cup \{r_i\}$
9:   end if
10: end for
11: return $(\pi_1, \pi_2)$
```

We now prove that Algorithm 1 is a $(1-\varepsilon)$-approximation algorithm for MAX-TNSW in our restricted setting. We need to show that the algorithm always returns in polynomial time two subsets π_1 and π_2 such that

$$u(\pi_1) \cdot u(\pi_2) \geq (1-\varepsilon)\,OPT \tag{3.2}$$

where OPT is the optimal value of the instance M.

Without loss of generality, we assume that $u(\pi_1) \geq u(\pi_2)$ and that r_j is the last resource that was assigned to agent a_1. This implies that $u(\pi_2) \geq u(\pi_1) - u(r_j)$. By addition of $u(\pi_2)$ to the two sides of the inequality we get

$$2u(\pi_2) \geq u(R) - u(r_j)$$

or

$$u(\pi_2) \geq \frac{1}{2}(u(R) - u(r_j)). \tag{3.3}$$

If $j \leq \ell$, it is easy to see that the obtained solution is indeed an optimal solution. Otherwise, we have $u(r_j) \leq u(r_i)$ for any $1 \leq i \leq \ell$, since the sequence $(r_1, r_2, \dots, r_m)$ was sorted in nonincreasing order according to their utilities. Therefore, one can easily check that

$$u(R) \geq (\ell + 1)u(r_j)$$

or, equivalently

$$\frac{u(r_j)}{u(R)} \leq \frac{1}{(\ell+1)}. \tag{3.4}$$

Furthermore, assume that $\pi^* = (\pi_1^*, \pi_2^*)$ is an optimal allocation for M, then due to the well-known inequality $a \cdot b \leq {}^{(a+b)^2}/_4$, we have:

$$OPT = u(\pi_1^*) \cdot u(\pi_2^*) \leq \frac{(u(\pi_1^*) + u(\pi_2^*))^2}{4} = \frac{(u(R))^2}{4}. \tag{3.5}$$

Using inequality (3.3) and (3.5) we get

$$\frac{u(\pi_1) \cdot u(\pi_2)}{OPT} \geq \frac{(u(\pi_2))^2}{\frac{(u(R))^2}{4}} \geq \frac{\left(\frac{u(R)}{2} - \frac{u(r_j)}{2}\right)^2}{\frac{(u(R))^2}{4}} = \left(1 - \frac{u(r_j)}{u(R)}\right)^2.$$

Now, using inequality (3.4) we obtain

$$\begin{aligned} u(\pi_1) \cdot u(\pi_2) &\geq \left(1 - \frac{1}{\ell + 1}\right)^2 OPT \\ &\geq (1 - \varepsilon) OPT. \end{aligned}$$

The running time of Algorithm 1 is dominated by the work in lines 1 and 3, since the other steps have a smaller cost. At line 1 of the algorithm, we need to sort the sequence of m resources, which takes time $\mathcal{O}(m \log m)$. Regarding line 3, we have to search exhaustively on the set of subsets of $\{r_1, \ldots, r_\ell\}$, and we need time $\mathcal{O}(2^\ell) \subseteq \mathcal{O}(1)$ (since ℓ is constant) to do this. In total, our algorithm runs in polynomial time, more precisely in time $\mathcal{O}(m \log m)$. This completes the proof. ❑

Similarly to the case of Max-TNSW, we have the following theorem. We omit the proof here since it is similar to the one of Theorem 3.5, except choosing $\ell = \lceil -1 + {}^1/_{(1-\varepsilon)} \rceil$.

Theorem 3.6 Max-ANSW *admits a PTAS when restricted to only two agents having the same additive utility functions.*

A PTAS for Max-ANSW for the case of unbounded identical agents

We now generalize the result in Theorem 3.6 to the case of an unbounded number of identical agents. Note that this restricted version is known to be strongly **NP**-hard and that obtaining an FPTAS is impossible, unless **P** = **NP**. However, we will show that one can get a PTAS for Max-ANSW based on the technique of rounding resources and using integer programming. The method can be seen as, and was largely inspired by, an adaptation of the methods due to Alon et al. [AAWY97, AAWY98]. Unfortunately, this method fails when applied to Max-TNSW.

Theorem 3.7 Max-ANSW *admits a PTAS for the case where all agents have the same additive utility functions.*

Proof. We first give an outline of the proof. Let ε be any fixed positive number less than 1. We will present a polynomial-time algorithm such that for any given instance M of Max-ANSW, it outputs an allocation π' for M such that $sw_{\overline{N}}(\pi') > (1-\varepsilon)OPT$, where OPT is value of an optimal allocation for M. The algorithm are divided into three stages: (1) Construct a new instance M^* from M by rounding the values of the single resources; (2) Find an optimal solution π^* for M^* using integer program; (3) Compute an allocation π' from π^* in polynomial time. These three stages of the algorithm are summarized in a diagram below.

$$\begin{tikzcd}
M = (A, R, u) \arrow[r, "(1)"] & M^* = (A, R^*, u^*) \arrow[d, "(2)"] \\
\pi' = (\pi'_1, \dots, \pi'_n) & \pi^* = (\pi^*_1, \dots, \pi^*_n) \arrow[l, "(3)"']
\end{tikzcd}$$

Let $M = (A, R, u)$ be a problem instance in which $A = \{a_1, \dots, a_n\}$ and $R = \{r_1, \dots, r_m\}$ are the sets of agents and resources, respectively, and agents have the same additive utility function $u > 0$. Let

$$L = \frac{1}{n} \cdot \sum_{j=1}^{m} u(r_j)$$

be the average utility of the agents. If there is some resource r_j with $u(r_j) \geq L$, there will be an optimal allocation for M in which r_j is the only single resource assigned to some agent a_p. Indeed, assume that there is an agent a_p whose bundle contains r_j and another resource r_s. That means $u(r_j + r_s) > L$, so there must be at least one agent a_q owning a bundle of value less than L. By assigning r_s again to agent a_q instead of a_p, it is easy to see that the new allocation is better than the old one, due to the property of the objective function. Therefore, it suffices to construct a PTAS for the case where every single resource has a value less than L. For the other cases, we will assign each single resource with a value greater than or equal to L to some agent, delete them all from the list, and then design a PTAS for the obtained instance.

Furthermore, we can also prove that if R does not contain any single resource of value greater than or equal to L, then there will be an optimal allocation $\pi = (\pi_1, \ldots, \pi_n)$ such that

$$\frac{L}{2} < u(\pi_i) < 2L$$

for all $i = 1, \ldots, n$.

First, if there exists an allocation in which agent a_i owns the bundle π_i with $u(\pi_i) \geq 2L$, one can obtain a better allocation by moving some single resource in π_i to another agent who owns a bundle of value less than L. Second, assume that $u(\pi_p) \leq {}^{L}\!/_{2}$ for some agent a_p. Then there must be an agent a_q owning the bundle π_q with $u(\pi_q) > L$. If π_q contains some single resource with value less than $u(\pi_q) - u(\pi_p)$, then by assigning this resource to agent a_p instead of a_q, we obtain a better allocation. Otherwise, agent a_q must own at least two single resources of value greater than $u(\pi_q) - u(\pi_p) > u(\pi_p)$. Then, by assigning one of these two resources to a_p and the bundle π_p to a_q, we will get again a better allocation.

Now we will show the method of rounding the values of the resources of M which leads to an easily solvable instance M^* (see [AAWY97, LMMS04]). Let α be some constant positive integer which depends on ε.

Construct instance $\mathbf{M}^* = (\mathbf{A}, \mathbf{R}^*, \mathbf{u}^*)$. The set of agents is the same as in M. For each resource $r_j \in R$ with $u(r_j) > {}^{L}\!/_{\alpha}$, create a corresponding resource r_j^* of value

$$u^*(r_j^*) = \frac{L}{\alpha^2} \cdot \left\lceil \frac{u(r_j)}{{}^{L}\!/_{\alpha^2}} \right\rceil.$$

Such a resource r_j^* is called a *big* resource. Moreover, for any big resource r_j^*, it is easy to prove that:

$$u(r_j) \leq u^*(r_j^*) \leq \frac{\alpha+1}{\alpha} \cdot u(r_j).$$

For the set T of all remaining resources $r_j \in R$ with $u(r_j) \leq {}^{L}\!/_{\alpha}$, let

$$S = \frac{L}{\alpha} \cdot \left\lceil \frac{u(T)}{{}^{L}\!/_{\alpha}} \right\rceil.$$

Create $S \cdot ({}^{\alpha}\!/_{L})$ new resources, each having a value of ${}^{L}\!/_{\alpha}$. These resources are called *small* resources. Let R^* be the set of all the big and small resources that have been created. Clearly, $\|R^*\| \leq m$. Let

$$L^* = \frac{1}{n} \sum_{r^* \in R^*} u^*(r^*).$$

One can check that $L \leq L^*$. Since each resource in R^* has a value of at most L (and so of at most L^*), we can show that there exists an optimal allocation $\pi^* = (\pi_1^*, \dots, \pi_n^*)$ for M^* such that

$$\frac{L^*}{2} < u^*(\pi_i^*) < 2L^*$$

for all i, $1 \leq i \leq n$, by using the same argument above for instance M.

Solve instance M*. Recall that each resource in R^* has the form of ${}^{z \cdot L}/_{\alpha^2}$ with $z \in \mathbb{N}$ and $\alpha \leq z \leq \alpha^2$. Hence, we can encode the information on which bundle is assigned to the agents as a vector $w = (w_\alpha, \dots, w_{\alpha^2})$, where w_z is the number of resources that have a value of $z \cdot {}^{L}/_{\alpha^2}$. The value of a bundle encoded by a vector w is given by

$$u^*(w) = \sum_{z=\alpha}^{\alpha^2} w_k \cdot z \cdot \frac{L}{\alpha^2}.$$

Note that there always exists an optimal solution in which each agent owns a bundle of value between ${}^{L^*}/_2$ and $2L^*$. Therefore, in order to find an optimal allocation for M^*, we will restrict our searching to the set of vectors w that satisfy this property. Let W be a set of such vectors w. Since each vector w contains at most 2α resources, it is easy to see that $\|W\|$ is bounded by a constant. The following integer program represents the instance M^* precisely.

$$\max \prod_{w \in W} \{u^*(w)\}^{\mu_w}$$

subject to

$$\begin{aligned} &\sum_{w \in W} \mu_w = n \\ &\sum_{w \in W} \mu_w \cdot w_z = n_z \quad \text{for} \quad z = \alpha, \dots, \alpha^2 \\ &\mu_w \geqslant 0 \qquad\qquad\quad \text{for all } w \in W, \end{aligned}$$

where μ_w denotes the number of agents owning the bundle w and n_z denotes the number of resources in R^* whose value equals $z \cdot ({}^{L}/_{\alpha^2})$.

The first constraint of the integer program above ensures that the number of agents is exactly n while the second one makes sure that all resources are assigned completely to the agents. Note that both the number of variables and the number of constraints are constant and thus the integer program above can be solved in polynomial time by using Lenstra's algorithm [Len83].

Turning to the last stage of the algorithm: we need to prove that given an instance M, one can find in polynomial time an allocation π' for M such that

$$sw_{\overline{N}}(\pi') > (1-\varepsilon)OPT$$

Assume that $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$ is an optimal allocation obtained by solving exactly M^*. We will construct in polynomial time an allocation $\pi' = (\pi_1', \ldots, \pi_n')$ for M from π^* such that

$$sw_{\overline{N}}(\pi') > \left(1 - \frac{5}{\alpha}\right) \cdot sw_{\overline{N}}(\pi^*). \tag{3.6}$$

Indeed, for every bundle π_i^*, we replace all the big resources r_j^* by the corresponding resources r_j and remove all the small resources. Denote by m_i the number of small resources in π_i^*. We then add the resources in R, each of value at most L/α, into π_i^* until the total value of the added resources exceeds $(m_i - 2)L/\alpha$. This step is always completed, since the fact that the following holds

$$\sum_{i=1}^{n}(m_i - 1) \cdot \frac{L}{\alpha} \leq S - n \cdot \frac{L}{\alpha} \leq u(T).$$

All the remaining resources (if they exist) can be assigned arbitrarily to the agents. It is easy to prove that the bundle π_i' obtained from π_i^* satisfies

$$u(\pi_i') \geq \frac{\alpha}{\alpha+1} \cdot u^*(\pi_i^*) - 2 \cdot \frac{L}{\alpha}.$$

Since $L \leq L^* < 2u^*(\pi_i^*)$, we have

$$u(\pi_i') > \left(1 - \frac{5}{\alpha}\right) \cdot u^*(\pi_i^*). \tag{3.7}$$

By taking the product of Equation (3.7) for $i = 1, \ldots, n$, and extracting the n root of both sides, we get Equation (3.6).

Now, we prove that

$$sw_{\overline{N}}(\pi^*) > \left(1 - \frac{2}{\alpha}\right) OPT. \tag{3.8}$$

To do this, we show that there exists a feasible solution $\pi'' = (\pi_1'', \ldots, \pi_n'')$ for M^* so that $sw_{\overline{N}}(\pi'') > (1 - 2/\alpha)\, OPT$. Let $\pi = (\pi_1, \ldots, \pi_n)$ be an optimal solution for M, then $OPT = sw_{\overline{N}}(\pi)$. For every bundle π_i, we replace the resources r_j that have a value greater than L/α by r_j^* and denote by s_i the total value of all remaining resources, collected in set $T \cap \pi_i$. Note that $\sum_{i=1}^{n} s_i = u(T)$. One can find easily n integers t_i such that $t_i \geq s_i - (L/\alpha)$ and $\sum_{i=1}^{n} t_i = S$. We then remove all resources, each of value at most L/α, and add $t_i \cdot (\alpha/L)$ resources of value L/α each. The new bundle π_i'' (obtained by modifying π_i) must satisfy $u^*(\pi_i'') \geq u(\pi_i) - (L/\alpha)$. Since $L < 2 \cdot u(\pi_i)$, it follows that

$$u^*(\pi_i'') > \left(1 - \frac{2}{\alpha}\right) \cdot u(\pi_i). \tag{3.9}$$

Again, by taking the product of Equation(3.9) for $i = 1, \ldots, n$, and extracting the n root of both sides, we get the following equation: $sw_{\overline{N}}(\pi'') > (1 - 2/\alpha) \cdot sw_{\overline{N}}(\pi)$. Note that $sw_{\overline{N}}(\pi^*) \geq sw_{\overline{N}}(\pi'')$, since π^* is an optimal allocation for M^*. Thus, we obtain Equation (3.8).

Finally, it follows from Equation (3.6) and Equation (3.8) that

$$\begin{aligned} sw_{\overline{N}}(\pi') &> \left(1 - \frac{2}{\alpha}\right) \cdot \left(1 - \frac{5}{\alpha}\right) \cdot OPT \\ &> \left(1 - \frac{7}{\alpha}\right) \cdot OPT. \end{aligned}$$

Choosing $\alpha = \lceil 7/\varepsilon \rceil$ completes the proof. ❑

An FPTAS for the case of a constant number of agents

We now consider the special case where the number of agents is not part of the input. For this case, we will design a fully polynomial-time approximation scheme for Max-ESW, Max-TNSW and Max-ANSW based on the dynamic programming method that was also used to give an FPTAS for a variety of scheduling problems (see [HS76] and [Sah76]). This will improve the results stated in Theorem 3.5 and Theorem 3.6.

Theorem 3.8 Max-ESW *with additive utilities admits an FPTAS for any fixed number of agents.*

Proof. Let $M = (A, R, U)$ be a problem instance where the number of agents n is bounded by a constant. As a shorthand, we denote by $s_{ij} = u_i(r_j)$ the utility of resource r_j for agent a_i for i, $1 \leq i \leq n$, and j, $1 \leq j \leq m$. Without loss of generality, we assume all s_{ij} to be nonnegative *integers.*

The proof of this theorem will be divided into two parts. In the first part, we construct a pseudo-polynomial time algorithm for Max-ESW that runs in time $\mathcal{O}(mB^n)$, where

$B = \max_{1 \le i \le n} u_i(R)$. We then prove in the second part that this algorithm yields a fully polynomial-time approximation scheme for our problem.

Let $T = (\vec{e}_1, \ldots, \vec{e}_n)$ be a canonical basis of the vector space $\mathbb{R}^n$, where $\vec{e}_i$ denotes the vector with a 1 in the i-th coordinate and 0's elsewhere. Now, consider Algorithm 2. We denote by V_j the set of all possible assignments of the first j resources to agents, where each assignment is encoded by an n-dimensional vector.

Algorithm 2 A pseudo-polynomial time algorithm for Max-ESW

```
1: $V_0 := \{\vec{0}\}$;
2: for $j := 1$ to $m$ do
3:     $V_j := \emptyset$;
4:     for each $\vec{v} \in V_{j-1}$ do
5:         $V_j := V_j \cup \{\vec{v} + s_{ij} \cdot \vec{e}_i \mid i = 1, \ldots, n\}$;
6:     end for
7: end for
8: return Vector $\vec{v} \in V_m$ such that the product of its coordinates is maximal;
```

More precisely, at step j of the loop in line 2 of Algorithm 2, every single vector $\vec{v} \in V_{j-1}$, which was created in the step $j-1$, will generate n vectors in V_j. It is easy to see that the number of vectors created for V_j will be $n \cdot \|V_{j-1}\|$ and thus increases exponentially. However, the size of every set V_j, $1 \le j \le m$, is bounded by $\mathcal{O}(B^n)$, since the coordinates of all vectors in V_j are integers not exceeding B. Hence, the running time of Algorithm 2 is $\mathcal{O}(m \sum_{k=1}^{m} \|V_k\|) = \mathcal{O}(mB^n)$, and thus it is a pseudo-polynomial time algorithm.

Now, coming to the second part of the proof, we make a small modification to the above pseudo-polynomial time algorithm in order to get an FPTAS. In more detail, we will remove some unnecessary vectors from V_j once they have been created. This will help to keep the number of vectors as low as possible. Of course, that may perhaps not return the exact optimal solutions but it will still give good approximations. A natural question is which vectors in V_j should be removed? First, we need to define an appropriate notion of equivalence of two n-dimensional vectors.

Let ε be any fixed positive number such that $0 < \varepsilon < 1$ and let α be a positive number depending on ε and m as follows:

$$\alpha = 1 + \frac{\varepsilon}{2m}.$$

Let $K = \lceil \log_\alpha B \rceil$ and $L_k = [\alpha^{p-1}, \alpha^p]$ for each p, $1 \le p \le K$. We define an equivalence relation $\sim$ on the set V_j as follows. Any two vectors $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_1, \ldots, y_n)$

are equivalent, denoted by $\vec{x} \sim \vec{y}$, if for every ℓ, $1 \leq \ell \leq n$, $x_\ell = y_\ell = 0$ or $x_\ell, y_\ell \in L_p$ for some p. Obviously, this relation is reflexive, symmetric, and transitive, and thus it is an equivalence relation. Moreover, under this relation, V_j 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 $\vec{x} \sim \vec{y}$ then

$$\frac{1}{\alpha} \cdot y_\ell \leq x_\ell \leq \alpha \cdot y_\ell \quad \text{or} \quad \frac{1}{\alpha} \cdot x_\ell \leq y_\ell \leq \alpha \cdot x_\ell \tag{3.10}$$

for all ℓ, $1 \leq \ell \leq n$. Indeed, the statement is obviously true if $x_\ell = y_\ell = 0$. Now, consider the case where $x_\ell, y_\ell \in L_p$ for some p, that is, $\alpha^{p-1} \leq x_\ell, y_\ell \leq \alpha^p$. In this case we have

$$\frac{x_\ell}{y_\ell} \geq \frac{\alpha^{p-1}}{\alpha^p} = \frac{1}{\alpha} \quad \text{and} \quad \frac{x_\ell}{y_\ell} \leq \frac{\alpha^p}{\alpha^{p-1}} = \alpha.$$

We are now ready to modify Algorithm 2 to obtain an FPTAS for the problem Max-ESW. We will do this by adding one more procedure, called Reduce(V_j), to Algorithm 2. The purpose of Reduce(V_j) is to reduce each set V_j once it has been created. More precisely, the set of vectors V_j is divided into equivalence classes according to the relation $\sim$ and then this procedure removes some vectors such that each class contains only one vector. The result is Algorithm 3.

Algorithm 3 An FPTAS for Max-ESW

```
1: $V_0^* := \{\vec{0}\}$;
2: **for** $j := 1$ **to** $m$ **do**
3:     $V_j := \emptyset$;
4:     **for** each $\vec{v}^* \in V_{j-1}^*$ **do**
5:         $V_j := V_j \cup \{\vec{v}^* + s_{ij} \cdot \vec{e}_i \mid i = 1, \ldots, n\}$;
6:     **end for**
7:     $V_j^*$ :=Reduce($V_j$);
8: **end for**
9: **return** Vector $\vec{v}^* \in V_m^*$ such that the product of its coordinates is maximal;
```

In Algorithm 3, we do the same steps as in Algorithm 2, but this time the set of vectors V_j will be created from V_{j-1}^* rather than from V_{j-1} and then is modified by the procedure Reduce applied to V_j to get the reduced set V_j^*. Note that the number of vectors in V_j^* is always bounded by $\mathcal{O}(K^n)$.

Now, we show the relationship between the two sets V_j and V_j^* for any j, $1 \leq j \leq m$. We will prove by induction on j that for every vector $\vec{v} = (v_1, \ldots, v_n) \in V_j$, there always exists a vector $\vec{v}^* = (v_1^*, \ldots, v_n^*) \in V_j^*$ such that

$$v_i^* \geq \frac{1}{\alpha^j} \cdot v_i$$

for all i, $1 \leq i \leq n$.

If $j = 1$, it is easy to see that $V_1^* = V_1$, hence the statement is obviously true. To prove the statement for every j, assume that the statement is true for $j-1$. Consider the set V_j and an arbitrary vector $\vec{v} = (v_1, \ldots, v_n)$ of V_j. This vector $\vec{v}$ must be created in line 5 of Algorithm 2 from some vector $\vec{w} = (w_1, \ldots, w_n)$ of the set V_{j-1}. Without loss of generality, we assume that $\vec{v}$ has the form of $(w_1 + s_{1j}, w_2, \ldots, w_n)$ (note that $v_1 = w_1 + s_{1j}$ and $v_i = w_i$ for all i, $2 \leq i \leq n$). Using the induction hypothesis above, there exists some vector $\vec{w}^* = (w_1^*, \ldots, w_n^*)$ in V_{j-1}^* such that

$$w_i^* \geq \frac{1}{\alpha^{j-1}} \cdot w_i$$

for all i, $1 \leq i \leq n$. On the other hand, note that $\vec{w}^* + s_{1j} \cdot \vec{e}_1 = (w_1^* + s_{1j}, w_2^*, \ldots, w_n^*)$ will also be created for V_j^* in line 5 of Algorithm 3, but it may be removed after line 7. However, there is another vector $\vec{v}^* = (v_1^*, \ldots, v_n^*) \in V_j^*$ such that $\vec{v}^* \sim (\vec{w}^* + s_{1j} \cdot \vec{e}_1)$. This yields

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

and for i, $2 \leq i \leq n$, if $w_i^* \neq 0$, we have

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

We now assume that Algorithm 2 returns a vector $\vec{v} = (v_1, \ldots, v_n) \in V_m$ such that $\min\{v_1, \ldots, v_n\} = OPT$ is maximal. Then, there must be a vector $\vec{v}^* = (v_1^*, \ldots, v_n^*) \in V_m^*$ such that

$$v_i^* \geq \frac{1}{\alpha^m} \cdot v_i$$

for all i, $1 \leq i \leq n$. This implies that

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

Moreover, we have

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

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

$$\max_{x\in[0,1]} f(x) = \max\{f(0), f(1), f(\ln 2)\} = f(0) = 0.$$

It follows that

$$\min\{v_1^*, \dots, v_n^*\} \ge \frac{1}{1+\varepsilon} OPT > (1-\varepsilon) OPT.$$

We next prove that Algorithm 3 has a running time that is polynomial in ${}^{1}\!/\!{}_{\varepsilon}$ and $|M|$, where $|M|$ denotes the size of M in some natural encoding. Indeed, the loop in Algorithm 3 needs m steps. At step j, the set V_j is created from V_{j-1}^* and this takes time $n \cdot \|V_{j-1}^*\| \in \mathcal{O}(n \cdot K^n)$ while the procedure Reduce(V_j) runs in time $\|V_j\|^2 \in \mathcal{O}(n^2 K^{2n})$. The overall time complexity of Algorithm 3 is in $\mathcal{O}(mK^{2n})$, since n is constant. Note further that

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

The above inequality can be proved as follows. Consider the function $f(a) = \ln a - 1 + {}^{1}\!/\!{}_{a}$ that is continuous and increasing on the interval $(1, \infty)$ 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 $|M| \ge \log B = (\log e)(\ln B)$. Thus, we have

$$K \le \left(1 + \frac{2m}{\varepsilon}\right) \frac{|M|}{\log e}.$$

This completes the proof. ❑

By changing slightly the parameter α, the method above can be applied well to derive an FPTAS for both Max-TNSW and Max-ANSW for the case of bounded number of agents. This is the best result we can obtain for this case.

Theorem 3.9 Max-TNSW *with the additive form admits an FPTAS when the number of agents is fixed.*

Proof. We use again the two algorithms described in the proof of Theorem 3.8. Every setting is unchanged except for choosing $\alpha = 1+\varepsilon/2nm$. We now assume that Algorithm 2 returns a vector $\vec{v} = (v_1, \ldots, v_n) \in V_m$ such that the product $\prod_{i=1}^{n} v_i = OPT$ is maximal. Then, there must be a vector $\vec{v}^* = (v_1^*, \ldots, v_n^*) \in V_m^*$ such that

$$v_i^* \geq \frac{1}{\alpha^m} \cdot v_i$$

for all i, $1 \leq i \leq n$. This implies that

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

Moreover, we have

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

and finally,

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

❑

Theorem 3.10 Max-ANSW *with the additive form admits an FPTAS when the number of agents is fixed.*

Proof. Choose $\alpha = 1 + \varepsilon/2m$, everything else is similar to the proof of Theorem 3.8 and of Theorem 3.9. ❑

An approximation algorithm for Max-TNSW and Max-ANSW in the general case

We now present a fast greedy approximation algorithm for both Max-TNSW and Max-ANSW. Our approach is based upon matching techniques for finding *maximum matchings* in certain classes of weighted bipartite graphs. Recall that a maximum matching (also known as *maximum cardinality matching*) is a matching with a maximum number of edges. For the weighted graphs, a *maximum weighted maximum matching* (MWMM for short) is a maximum matching such that the sum of the weights of the edges in the matching is maximum. It is shown that a problem of finding a MWMM on weighted bipartite graphs can be solved in polynomial time (see [MN99]). A variant of this problem, which is defined below, is to find a maximum matching that maximizes the product of the weights of the edges in the matching.

Maximum Product Weighted Matching (MPWM)	
Input:	A weighted bipartite graph $G = (X \cup Y, E)$ and a weight function $w : E \to \mathbb{R}^+$.
Output:	A maximum matching $\mathcal{M} \subseteq E$ that maximizes $\prod_{e \in \mathcal{M}} w(e)$.

In the following we will prove that this variant can be also solved in polynomial time.

Lemma 3.1 *Given a weighted bipartite graph G, a* MPWM *of G can be found in polynomial time.*

Proof. We may assume that G is a complete bipartite graph, since if there exist two vertices $x, y \in V$ with $\{x, y\} \notin E$, we can add the edge $\{x, y\}$ of weight zero to E. Moreover, by multiplying every weight $w(e)$ by a large enough number, it suffices to consider the case when $w(e) \in \mathbb{R}_{\geq 1} \cup \{0\}$. The basic idea is to transform the given MPWM instance into an instance of the problem of finding a maximum weighted maximum matching, which can be solved in polynomial time.

The weight function $w' : E \to \mathbb{R}$ is defined as follows:

$$w'(e_i) \;=\; \begin{cases} \log w(e_i) & \text{if } w(e_i) \neq 0 \\ -z \log \Delta & \text{otherwise,} \end{cases}$$

where $\Delta = \max_{e_i \in E} w(e_i)$ and $z = \min\{\|X\|, \|Y\|\}$. Without loss of generality, we can assume that $\mathcal{M} = \{e_1, \ldots, e_z\}$ is a maximum matching of G with weight function w' so that $\sum_{i=1}^{z} w'(e_i)$ is maximal. We now prove that $\mathcal{M}$ is exactly a matching of G that maximizes $\prod_{i=1}^{z} w(e_i)$. It is easy to see that $\sum_{i=1}^{z} w'(e_i) \leq 0$ if and only if there exists an edge $e_i \in \mathcal{M}$ with negative weight $w'(e_i) = -z \log \Delta$, i.e., $w(e_i) = 0$. In this case, for every maximum matching $\mathcal{M}$ of G, the product of the edge weights in $\mathcal{M}$ is zero with respect to the weight function w. Now suppose that $w(e_i) > 0$ for all $i \in \{1, \ldots, z\}$. Assume that there exists another matching $\mathcal{M}' = \{e'_1, \ldots, e'_z\}$ of G such that

$$\prod_{i=1}^{z} w(e'_i) > \prod_{i=1}^{z} w(e_i).$$

This implies that

$$\log\left(\prod_{i=1}^{z} w(e_i')\right) > \log\left(\prod_{i=1}^{z} w(e_i)\right)$$

or, equivalently,

$$\sum_{i=1}^{z} \log w(e_i') > \sum_{i=1}^{z} \log w(e_i),$$

which in turn is equivalent to

$$\sum_{i=1}^{z} w'(e_i') > \sum_{i=1}^{z} w'(e_i).$$

This is a contradiction. ❑

We are now ready to give an approximation algorithm for both Max-TNSW and Max-ANSW, assuming that utility functions are additive. Our proof of Theorem 3.11 transforms the problem in to a problem of finding a MPWM in certain bipartite graph. Particularly, our approach shows that the lower bound results for Max-ANSW and Max-TNSW in Theorem 3.11 and Theorem 3.12 remain valid even if the additive utility functions are replaced by the submodular utility functions.

Theorem 3.11 Max-ANSW *can be approximated within a factor of* $1/(m-n+1)$ *when agents' utility functions are in the additive form.*

Proof. Let M be an instance of Max-TNSW with n agents $A = \{a_1, \dots, a_n\}$ and m resources $R = \{r_1, \dots, r_m\}$, and assume each agent has an individual value for each resource, say $u_i(r_j)$. In the following algorithm we denote by μ_i the bundle assigned to agent a_i, for $1 \le i \le n$.

Obviously, the Algorithm 4 computes an incomplete allocation $\mu = (\mu_1, \dots, \mu_n)$ for the instance M in polynomial time. For convenience, re-index (if necessary) such that $\mu_i = \{r_i\}$ for all $i, 1 \le i \le n$. We can lift μ to a complete allocation X by allocating the remaining resources $r_j, j \ne i$, which is not assigned by Algorithm 4 yet, arbitrarily to agents. Clearly, we have

$$sw_{\overline{N}}(X) \ge \left(\prod_{i=1}^{n} u_i(r_i)\right)^{1/n}.$$

Algorithm 4 An approximation algorithm for Max-TNSW and Max-ANSW.

```
1:  for $i := 1$ to $n$ do
2:      $\mu_i := \emptyset$;
3:  end for
4:  $X := A$; $Y := R$;
5:  $E := \{(a_i, r_j) | \, a_i \in A, r_j \in R, 1 \leq i \leq n, \, 1 \leq j \leq m\}$;
6:  $G := (X \cup Y, E)$;
7:  for each edge $(a_i, r_j) \in E$ do
8:      $w((a_i, r_j)) := u_i(r_j)$;
9:  end for
10: Finding a Maximum-Product-Weighted-Matching $\mathcal{M}$ of $G$;
11: for each $(a_i, r_j) \in \mathcal{M}$ do
12:     $\mu_i := \{r_j\}$;
13: end for
14: return $\mu = (\mu_1, \ldots, \mu_n)$;
```

Let $(\pi_1, \ldots, \pi_n)$ be an optimal allocation for M. We will prove that:

$$\left(\prod_{i=1}^{n} u_i(r_i)\right)^{1/n} \geq \frac{1}{(m-n+1)} \left(\prod_{i=1}^{n} u_i(\pi_i)\right)^{1/n}.$$

For every bundle π_i, denote by r_i^* the single resource of biggest value, that is, $u_i(r_i^*) = \max_{r \in \pi_i} u_i(r)$. We have

$$\frac{1}{(m-n+1)} \left(\prod_{i=1}^{n} u_i(\pi_i)\right)^{1/n} \leq \frac{1}{(m-n+1)} \left(\prod_{i=1}^{n} \{\lambda_i \cdot u_i(r_i^*)\}\right)^{1/n},$$

where λ_i is the number of single resources belonging to π_i, for all i, $1 \leq i \leq n$. The right hand side of the inequality above can be transformed into the following form:

$$\frac{(\lambda_1 \cdots \lambda_n)^{1/n}}{(m-n+1)} \left(\prod_{i=1}^{n} u_i(r_i^*)\right)^{1/n} \leq \frac{(\lambda_1 \cdots \lambda_n)^{1/n}}{(m-n+1)} \left(\prod_{i=1}^{n} u_i(r_i)\right)^{1/n},$$

since the set of edges $\{(a_1, r_1^*), \ldots, (a_n, r_n^*)\}$ is a maximum matching of G, whereas $\{(a_1, r_1), \ldots, (a_n, r_n)\}$ is a matching of G with maximum product of edge weights.

Finally, note that $\sum_{i=1}^{n} \lambda_i = m$, and by the well-known arithmetic-geometric mean inequality, we have

$$\frac{(\lambda_1 \cdots \lambda_n)^{1/n}}{m-n+1} \leq \frac{\lambda_1 + \cdots + \lambda_n}{n \cdot (m-n+1)} = \frac{m}{n \cdot (m-n+1)} \leq 1.$$

The last inequality above follows from the fact that $n \leq m$. ❑

By applying the Algorithm 4 again, we can show the similar result for MAX-TNSW.

Theorem 3.12 MAX-TNSW *with the additive form can be approximated to a factor of* $1/(m-n+1)^n$.

A special case with $n = m$

Although the problem of maximizing Nash social welfare is computationally hard in general for the additive form, there are some interesting naturally restricted instances where it is computationally easy. Here we consider a restricted version of this problem where there are as many agents as resources. We present an exact polynomial-time algorithm for MAX-TNSW and MAX-ANSW by converting these problems to the problem of finding a maximum matching in some complete bipartite graph. In fact, the results can be extended easily to the general case of the k-additive form for any fixed $k \geq 1$.

Theorem 3.13 *For additive utilities,* MAX-TNSW *and* MAX-ANSW *can be solved exactly in polynomial time when the number of agents and the number of resources are the same.*

Proof. It is important to note that each of the agent will get only one single resource, as otherwise both the total and average Nash social welfare would be zero. Let $M = (A, R, U)$ be an instance of the problems MAX-TNSW and MAX-ANSW. An optimal allocation π for M will correspond exactly to a MAXIMUM-PRODUCT-WEIGHTED-MATCHING $\mathcal{M}$ of a suitable graph G that is constructed from M. Hence, such an optimal allocation π could be found in polynomial time by using Algorithm 4. ❑

3.4 Conclusion and Future Work

In this chapter, we have studied several typical social welfare maximization problems raising in the field of multiagent resources allocation. The results obtained in this chapter improve our theoretical understanding of the approximability of these problems, both in terms of upper and lower bounds (see Table 3.5 for a summary of the results).

For the bundle form and the 3-additive form, we have shown that, unless **P** = **NP**, there is no hope for finding polynomial approximation algorithms to within any factor for Max-ESW, Max-TNSW and Max-ANSW. In addition, we have established the constant hardness results for the problem of maximizing Nash social welfare in the additive form. In terms of approximability, we have investigated scenarios where agents use additive utility functions to present their preferences. We have designed a fast greedy approximation algorithm for both Max-TNSW and Max-ANSW. Note that this result even holds for the case of submodular utility functions. Additionally, we have achieved a fully polynomial-time approximation scheme for Max-ESW, Max-TNSW and Max-ANSW under the assumption that the number of agents is bounded by a constant. This assumption is actually very natural (it is more likely that we will have to share a huge number of resources among a small set of agents than vice versa), so this almost comes down to a FPTAS result. Regarding the case where the number of agents is not bounded, we have shown that Max-ANSW admits a PTAS if all agents have the same utility functions.

For the future work, a challenging open problem is to devise a (constructive) constant approximation algorithm for Max-ESW with respect to the additive form. Note that a ($1/4$)-approximation algorithm for this problem presented in [AFS12] does not provide the way to find the appropriate solution in polynomial time. Another interesting question is for the problem Max-ANSW: how to narrow the gap between the approximation ratio of $1/(m-n+1)$ presented here and the factor $2\sqrt{2}/3$ hardness result. The same open question is applied to Max-TNSW. Finally, it would be also interesting to improve a PTAS for Max-TNSW with an unbounded number of agents having the same additive utility functions.

Table 3.5: New results for social welfare optimization problems with respect to the bundle form and the k-additive form.

Inapproximability

Social Welfare	Bundle	Reference	k-Additive	Reference
Max-USW	$m^{\varepsilon-1/2}$	[LOS99]	$(21/22)+\varepsilon,\ k \geq 2$	Pro. 3.1
Max-ESW	any factor	Thm. 3.1	any factor, $k \geq 3$	Thm. 3.4
			$(1/2)+\varepsilon,\ k = 1$	[BD05]
Max-TNSW	any factor	Thm. 3.1	any factor, $k \geq 3$	Thm. 3.4
			$(8/9)+\varepsilon,\ k \in \{1,2\}$	Thm. 3.2
Max-ANSW	any factor	Thm. 3.1	any factor, $k \geq 3$	Thm. 3.4
			$\left(2\sqrt{2}/3\right)+\varepsilon,\ k \in \{1,2\}$	Thm. 3.3

Approximability (additive form)

Social Welfare	general	Ref.	$n =$ const.	Ref.	id. agent	Ref.
Max-ESW	$1/\sqrt{n}\log^3 n$	[AS10]	FPTAS	Thm. 3.8	PTAS	[Woe97]
Max-TNSW	$1/(m-n+1)^n$	Thm. 3.12	FPTAS	Thm. 3.5	open	– –
Max-ANSW	$1/m-n+1$	Thm. 3.11	FPTAS	Thm. 3.10	PTAS	Thm. 3.7

Exact polynomial algorithm (additive form)

Social Welfare	$m = n$	Reference
Max-ESW	poly. time	[Gol05]
Max-TNSW	poly. time	Thm. 3.13
Max-ANSW	poly. time	Thm. 3.13

Chapter 4

Computing Minimum Envy Allocation

In this chapter, we study the problem of fairly allocating a set of goods among several agents that are assumed to have additive utility functions over bundles of goods. As a most prominent interpretation of fairness, we focus on envy-freeness, which means that no agent wants to swap her bundle of goods in an allocation with another agent. Much of the work so far has been devoted to envy-freeness in the setting where one divisible good is to be divided among the agents (the so-called *cake-cutting* problem, see [BT96, RW98]). Here, we focus on computing the envy-free allocation of indivisible goods. Let $\pi = (\pi_1, \ldots, \pi_n)$ be an allocation amongst n agents, where π_i is the bundle assigned to agent a_i, and let u_i be a utility function of agent a_i, $1 \leq i \leq n$, an envy-free allocation is defined as follows:

Definition 4.1 *An allocation π is said to be envy-free if and only if for any agent a_i, the following inequality holds for any agent a_j:*

$$u_i(\pi_i) \geq u_i(\pi_j)$$

Example 4.1 *Consider an instance with 3 agents and 7 single goods, and agents' utility functions have additive form which are given in Table 4.1. The numbers in boldface show an allocation of goods that is envy-free: agent a_1 gets the bundle $\{r_4, r_6\}$, agent a_2 gets the bundle $\{r_1, r_7\}$ and agent a_3 gets the last three goods $\{r_2, r_3, r_5\}$.*

In fact, while envy-free allocations of a divisible good always exist (and can even be guaranteed by a finite bounded procedure [BT95]), an envy-free allocation of indivisible goods may not exist in general (see Example 4.2). Therefore, we seek to compute allocations that ensure envy to be as small as possible. There are several ways to define a measure of envy for a given allocation. Chevaleyre et al. [CEEM07] proposed a

Table 4.1: Utilities of the agents for Example 4.1.

Resources	Agent a_1	Agent a_2	Agent a_3
r_1	0	**5**	3
r_2	3	1	**2**
r_3	3	1	**4**
r_4	**5**	2	0
r_5	2	4	**3**
r_6	**4**	3	5
r_7	2	**4**	5

framework for defining the degree of envy of an allocation based on the degree of envy among individual agents. Agent a_i's envy regarding agent a_j's bundle is determined by $u_i(\pi_j) - u_i(\pi_i)$, where u_i is a_i's utility function, and a_i's envy with respect to allocation π is defined by using the aggregation functions max ($\max_{j \neq i}\{u_i(\pi_j) - u_i(\pi_i)\}$) and sum ($\sum_{j \neq i}(u_i(\pi_j) - u_i(\pi_i))$). Finally, the envy of the allocation π is aggregated from the envy of individual agents via the aggregation functions max and sum. Considering the optimization problems based on this measure of envy, a drawback of this approach is that, unless $\mathbf{P} = \mathbf{NP}$, there are no approximation algorithms for them, since the objective function might be zero (see the work of Lipton et al. [LMMS04]). We circumvent this by defining similar notions of degree of envy based on max and product ($\prod_{j \neq i} {}^{u_i(\pi_j)}/_{u_i(\pi_i)}$), and study approximability of the corresponding optimization problems. The results presented in this chapter will appear in [NR].

Related work: A number of papers in the literature has investigated the computation of envy-free allocations for indivisible goods. These papers focus on either centralized or distributed protocols, or take into account the way in which the agents' preferences are expressed: cardinal or ordinal. Chevaleyre et al. [CEEM07] studied a distributed protocol in which agents negotiate on the exchange of goods to reach an allocation that is envy-free or has minimal envy. Regarding centralized protocols, Bouveret, Endriss, and Lang [BEL10] dealt with the problem where the agents' preferences are represented ordinally by using so-called SCI-nets, while Bouveret and Lang [BL08] considered the logical aspects of representation and related complexity issues. Lipton et al. [LMMS04] addressed the problem of computing allocations with minimal envy when agents have numerical additive preferences, which corresponds exactly to our problem of minimizing $er^{\max,\max}$ (which will be defined in Section 4.1). Among other results, they provided a PTAS for the case of agents with identical utility functions and mentioned that one can obtain even a fully polynomial-time approximation scheme (FPTAS) for this case if the number of agents is fixed, thus extending the corresponding result of Bazgan et al. [BST98] for the problem Subset-Sums Ratio: given a set $S = \{c_1, \ldots, c_n\}$ of positive integers, the goal is to find a partition of S into two subsets S_1, S_2 with $\sum_{c_i \in S_1} c_i \geq \sum_{c_i \in S_2} c_i$ such that the ratio $\sum_{c_i \in S_1} c_i / \sum_{c_i \in S_2} c_i$ is minimized. However, their method that gave an FPTAS for the problem cannot be applied for the case of agents with different preferences.

4.1 Degree of Envy

Let $A = \{a_1, \ldots, a_n\}$ be a set of agents and $R = \{r_1, \ldots, r_m\}$ be a set of indivisible goods. Each agent a_i has an additive utility function $u_i : 2^R \to \mathbb{Q}^+$, i.e., for every subset $B \subseteq R$, $u_i(B) = \sum_{r_j \in B} u_i(r_j)$. An allocation is a partition π of R into n subsets $(\pi_1, \ldots, \pi_n)$, where π_i is assigned to agent a_i. We now adopt the approach of Chevaleyre et al. [CEEM07], who introduced the notion of *degree of envy* of society. Their definition proceeds in three stages by first defining envy between any two agents, then envy of any agent with respect to all other agents, and finally envy of society. With respect to a given allocation $\pi = (\pi_1, \ldots, \pi_n)$, the three stages of the process are as follows:

- **Stage 1 (envy between any two agents):** For each i and $j \neq i$, *agent a_i's envy with respect to agent a_j* is defined as the ratio between a_i's utility for the bundle assigned to a_j and a_i's utility for the bundle assigned to herself:

$$er(i, j) = \frac{u_i(\pi_j)}{u_i(\pi_i)}.$$

- **Stage 2 (degree of envy of any agent):** Here we measure how envious any agent a_i is with respect to anyone else, where we consider two aggregation functions, product and max:

$$er^{\text{pro}}(i) = \prod_{j \neq i} er(i, j) \qquad \text{and} \qquad er^{\max}(i) = \max_{j \neq i} er(i, j).$$

- **Stage 3 (degree of envy of society):** Based on the degree of envy of individual agents, we define the *degree of envy of society* for allocation π, by again considering the two aggregation functions max and product. While the max function focuses on the most envious agent of society, the product measures envy of society as a whole:

$$er^{\text{pro,opt}}(\pi) = \prod_{i=1}^{n} er^{\text{opt}}(i) \quad \text{and} \quad er^{\text{max,opt}}(\pi) = \max_{i=1}^{n} er^{\text{opt}}(i),$$

where opt $\in \{\max, \text{pro}\}$.

Here we only consider these two operators, *max* and *product*, for aggregating degrees of envy of individual agents. However, there are also other potential alternatives for the aggregation of individual preferences, e.g., using the *leximin ordering* (see Moulin [Mou88]).

Note that one of the two measures of envy of society given by Lipton et al. [LMMS04] corresponds to $er^{\max,\max}(\pi)$.

Now, given these notions of the degree of envy of society, we define the following optimization problems, where we let $\text{opt}_1, \text{opt}_2 \in \{\max, \text{pro}\}$.

Minimum Envy $(\text{opt}_1, \text{opt}_2)$	
Input:	A set of m indivisible goods and a set of n agents, each having an additive utility function over the bundles of goods.
Output:	An allocation π that minimizes $\max\{1, er^{\text{opt}_1,\text{opt}_2}(\pi)\}$.

Note that we minimize $\max\{1, er^{\text{opt}_1,\text{opt}_2}(\pi)\}$ rather than $er^{\text{opt}_1,\text{opt}_2}(\pi)$ in order to make the problem fit with the common definition of objective functions in optimization problems so that approximation algorithms can be applied to it and analyzed in a standard manner.

Example 4.2 *Consider an instance with three agents and six single goods, where each agent has an additive utility function which is given in Table 4.2.*

Table 4.2: Utilities of the agents for Example 4.2.

Resources	Agent a_1	Agent a_2	Agent a_3
r_1	1	**5**	5
r_2	2	5	**4**
r_3	0	0	**1**
r_4	3	1	**6**
r_5	**4**	1	4
r_6	**3**	0	2

There is no envy-free allocation for this instance. The numbers in boldface show an allocation whose envy is minimized: $\pi = (\{r_5, r_6\}, \{r_1\}, \{r_2, r_3, r_4\})$ *with*

$$er^{\max,\max}(\pi) = \left\{\frac{1}{7}, \frac{5}{7}, \frac{1}{5}, \frac{6}{5}, \frac{6}{11}, \frac{5}{11}\right\} = \frac{6}{5}.$$

In comparison with other notions of fairness, it is worth noting that allocations with minimum envy do not always optimize either egalitarian social welfare (the utility of the agent who is worst off) or social welfare by Nash (the product of the individual agent utilities), and vice versa. For instance, the allocation π shown in Example 4.2 has the Nash product $7 \cdot 5 \cdot 11 = 385$ and thus does not maximize Nash social welfare. Also, it is not an optimal allocation with respect to the egalitarian social welfare. Conversely, the allocation $\pi' = (\{r_5, r_6\}, \{r_1, r_2\}, \{r_3, r_4\})$ maximizes both the egalitarian and Nash social welfare, but its envy $er^{\max,\max}(\pi') = 9/7$ is not minimal.

4.2 Our Results

4.2.1 Approximation Schemes

In this section, we prove that there is a fully polynomial time approximation scheme (FPTAS) for the problem of minimizing envy, for any pair $(\mathrm{opt}_1, \mathrm{opt}_2)$, where $\mathrm{opt}_i \in \{\max, \mathrm{pro}\}$, and for a bounded number of agents. The technique is again similar to that used in designing an FPTAS for the problem of maximizing social welfare in the previous chapter.

Let $M = (A, R)$ be an instance of the problem in which each agent a_i, $1 \leq i \leq n$, has an additive utility function over the power set 2^R. We denote by s_{ij} the value of good r_j for agent a_i. Without loss of generality, we assume that $s_{ij} \in \mathbb{N}$ for all i and j. In Algorithm 5 below, V_j denotes the set of all possible allocations of $\mathcal{R}_j = \{r_1, \ldots, r_j\}$, which assigns the first j goods in R to agents. Each allocation in V_j is represented by a vector $\vec{v} = (v_{ip})$ in which v_{ip} is an evaluation of a_p's bundle by agent a_i for all $i, p \in \{1, \ldots, n\}$. The envy of the allocation corresponding to a vector $\vec{v} \in V_j$ is denoted by $er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v})$. We denote by $\vec{\mu}_{k,i}$, $1 \leq k, i \leq n$, a vector of dimension n^2 with a 1 in the coordinate t_{ki} and a 0 everywhere else. Note that there are totally n^2 such vectors.

Algorithm 5 A pseudo-polynomial time algorithm for MINIMUM ENVY $(\mathrm{opt}_1, \mathrm{opt}_2)$

```
1: $V_0 := \{\vec{0}\}$;
2: for $j := 1$ to $m$ do
3:   $V_j := \emptyset$;
4:   for each $\vec{v} \in V_{j-1}$ do
5:     for $i := 1$ to $n$ do
6:       $V_j := V_j \cup \{\vec{v} + \sum_{k=1}^{n} s_{kj} \cdot \vec{\mu}_{k,i}\}$
7:     end for
8:   end for
9: end for
10: return vector $\vec{v} \in V_m$ that minimizes $er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v})$.
```

Let $B = \max_{1 \le i \le n} \sum_{j=1}^{m} s_{ij}$. By using the same argument as in the proof of Theorem 3.8, one can show that the running time of Algorithm 5 is $\mathcal{O}(mB^{n^2})$ (note that each vector in the set V_j has dimension of n^2) and thus it is a pseudo-polynomial time algorithm.

Let ε be any fixed number such that $0 < \varepsilon < 1$, and

$$\alpha = 1 + \frac{\varepsilon}{4m\delta}$$

where δ depends on n and will be determined later. Let $K = \lceil \log_\alpha B \rceil$ and $L_k = [\alpha^{k-1}, \alpha^k]$, where $1 \le k \le K$. We define the equivalence relation $\sim$ on the set V_j as same as before and we also have that if $x \sim y$ then

$$\frac{1}{\alpha} \cdot y_\ell \le x_\ell \le \alpha \cdot y_\ell \quad \text{and} \quad \frac{1}{\alpha} \cdot x_\ell \le y_\ell \le \alpha \cdot x_\ell \tag{4.1}$$

for all ℓ, $1 \le \ell \le n^2$.

Algorithm 6 An FPTAS for MINIMUM ENVY $(\mathrm{opt}_1, \mathrm{opt}_2)$

```
1: $V_0^* := \{\vec{0}\}$;
2: for $j := 1$ to $m$ do
3:   $V_j := \emptyset$;
4:   for each $\vec{v^*} \in V_{j-1}^*$ do
5:     for $i := 1$ to $n$ do
6:       $V_j := V_j^* \cup \{\vec{v^*} + \sum_{k=1}^{n} s_{kj} \cdot \vec{\mu}_{k,i}\}$
7:     end for
8:   end for
9:   $V_j^* :=$ REDUCE$(V_j)$
10: end for
11: return vector $\vec{v^*} \in V_m^*$ that minimizes $er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v^*})$.
```

In the following, we will show the relationship between the two sets V_j and V_j^*.

Lemma 4.1 *Let V_j and V_j^* be the two sets of vectors that have been created by Algorithms 5 and 6, respectively. For each vector $\vec{v} = (v_{ip}) \in V_j$, there always exists a vector $\vec{v^*} = (v^*_{ip}) \in V_j^*$ such that for all $i \neq p, 1 \leq i, p \leq n$:*

$$v^*_{ii} \geq \frac{1}{\alpha^j} \cdot v_{ii} \quad \textit{and} \quad v^*_{ip} \leq \alpha^j \cdot v_{ip}.$$

Proof. The proof is by induction on j. The case with $j = 1$ is true by Equation (4.1) and the fact that $V_1^* = V_1$. Assume that the statement is true for $j-1$. Consider an arbitrary vector $\vec{v} = (v_{ip}) \in V_j$. This vector $\vec{v}$ must be created in line 6 of Algorithm 5 from some vector $\vec{w} = (w_{ip})$ of the set V_{j-1}. Without loss of generality, we assume that $\vec{v}$ has the form

$$(w_{11} + s_{1j}, w_{12}, \ldots, w_{1n}, \ldots, w_{n1} + s_{nj}, w_{n2}, \ldots, w_{nn}),$$

where $v_{i1} = w_{i1} + s_{ij}$ for $i \in \{1, \ldots, n\}$ and $v_{ip} = w_{ip}$ for $p \neq 1$. Using the inductive assumption above, there exists some vector $\vec{w^*} = (w^*_{ip}) \in V^*_{j-1}$ such that

$$w^*_{ii} \geq \frac{1}{\alpha^{j-1}} \cdot w_{ii} \quad \text{and} \quad w^*_{ip} \leq \alpha^{j-1} \cdot w_{ip} \quad (i \neq p) \tag{4.2}$$

for all $i, p \in \{1, \ldots, n\}$. On the other hand, note that the vector

$$(w^*_{11} + s_{1j}, w^*_{12}, \ldots, w_{1n}, \ldots, w^*_{n1} + s_{nj}, w^*_{n2}, \ldots, w^*_{nn})$$

will also be created for V_j but may be removed by the procedure $\textsc{Reduce}(V_j)$, which outputs V_j^*. However, there must be another vector $\vec{v^*} = (v^*_{ip}) \in V_j^*$ such that $\vec{v^*} \sim \vec{w^*}$. This yields

$$v^*_{11} \geq \frac{1}{\alpha} \cdot (w^*_{11} + s_{1j}) \geq \frac{1}{\alpha^j} \cdot w_{11} + \frac{1}{\alpha} \cdot s_{1j} \geq \frac{1}{\alpha^j} \cdot (w_{11} + s_{1j}) = \frac{1}{\alpha^j} \cdot v_{11}.$$

For $i \neq 1$, we have

$$v^*_{ii} \geq \frac{1}{\alpha} \cdot w^*_{ii} \geq \frac{1}{\alpha^j} \cdot w_{ii} = \frac{1}{\alpha^j} \cdot v_{ii}$$

and

$$v^*_{i1} \leq \alpha \cdot (w^*_{i1} + s_{ij}) \leq \alpha^j \cdot w_{i1} + \alpha \cdot s_{ij} \leq \alpha^j \cdot (w_{i1} + s_{ij}) = \alpha^j \cdot v_{i1}.$$

For any $p \neq 1$ and $i \neq p$, we have

$$v^*_{ip} \leq \alpha \cdot w^*_{ip} \leq \alpha^j \cdot w_{ip} = \alpha^j \cdot v_{ip}.$$

❑

We are now ready to prove the main result of this section.

Theorem 4.1 *For any fixed* $\varepsilon > 0$ *and any pair* $(\mathrm{opt}_1, \mathrm{opt}_2)$ *with* $\mathrm{opt}_i \in \{\max, \mathrm{pro}\}$*, Algorithm 6 always produces in polynomial time an allocation whose envy is within a factor of* $1 + \varepsilon$ *of the optimum.*

Proof. Let $\vec{v} = (v_{ip}) \in V_m$ be a vector returned by Algorithm 5. By Lemma 4.1, there exists a vector $\vec{v^*} = (v^*_{ip}) \in V^*_m$ such that

$$v^*_{ii} \geq \frac{1}{\alpha^m} \cdot v_{ii} \quad \text{and} \quad v^*_{ip} \leq \alpha^m \cdot v_{ip} \quad (i \neq p)$$

for all $i, p \in \{1, \ldots, n\}$. Assume Algorithm 6 outputs a vector $\vec{x}$. By Lemma 4.1 and the fact that

$$er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v^*}) \geq er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{x}) \geq er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v}),$$

it is easy to see that $er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{x}) = \infty$ if and only if $er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v}) = \infty$. In case $er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v}) < \infty$, we have $v_{ii}, v^*_{ii} \neq 0$ for all i, and so

$$\frac{1}{v^*_{ii}} \leq \alpha^m \cdot \frac{1}{v_{ii}}$$

and thus,

$$\frac{v^*_{ip}}{v^*_{ii}} \leq \alpha^{2m} \cdot \frac{v_{ip}}{v_{ii}} \quad (i \neq p)$$

for all $i, p \in \{1, \ldots, n\}$. By choosing δ appropriately, we can show that for each pair $(\mathrm{opt}_1, \mathrm{opt}_2)$:

$$\max\{1, er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{x})\} \leq (1 + \varepsilon) \max\left\{1, er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v})\right\}. \tag{4.3}$$

Indeed, we prove the claim for the four possible cases below. Note that we have $er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{x}) \leq er^{\mathrm{opt}_1,\mathrm{opt}_2}(\vec{v^*})$.

Note that we have already proved the following inequality in the previous chapter:

$$\lambda^{2m\delta} = \Big(1 + \frac{\varepsilon}{4m\delta}\Big)^{2m\delta} \leq e^{\varepsilon/2} \leq 1 + \varepsilon. \tag{4.4}$$

Case 1: $(\mathrm{opt}_1, \mathrm{opt}_2) = (\mathrm{pro}, \mathrm{pro})$. In this case, we have

$$\begin{aligned}\max\{1, er^{\mathrm{pro,pro}}(\vec{x})\} \leq\ & \max\left\{1, \prod\nolimits_{i=1}^{n} \prod\nolimits_{p\neq i} \frac{v_{ip}^*}{v_{ii}^*}\right\} \\ \leq\ & \max\left\{1, \alpha^{2mn(n-1)} \prod\nolimits_{i=1}^{n} \prod\nolimits_{p\neq i} \frac{v_{ip}}{v_{ii}}\right\} \\ \leq\ & \alpha^{2mn(n-1)} \max\{1, er^{\mathrm{pro,pro}}(\vec{v})\}.\end{aligned}$$

Choosing $\delta = n(n-1)$ and applying inequality (4.4), we obtain (4.3).

Case 2: $(\mathrm{opt}_1, \mathrm{opt}_2) = (\max, \mathrm{pro})$. Here we obtain

$$\begin{aligned}\max\{1, er^{\mathrm{max,pro}}(\vec{x})\} \leq\ & \max\nolimits_{i=1}^{n} \left\{1, \prod\nolimits_{p\neq i} \frac{v_{ip}^*}{v_{ii}^*}\right\} \\ \leq\ & \max\nolimits_{i=1}^{n} \left\{1, \alpha^{2m(n-1)} \prod\nolimits_{p\neq i} \frac{v_{ip}}{v_{ii}}\right\} \\ \leq\ & \alpha^{2m(n-1)} \max\left\{1, er^{\mathrm{max,pro}}(\vec{v})\right\}.\end{aligned}$$

Choosing $\delta = n-1$ and applying inequality (4.4), we again obtain (4.3).

Case 3: $(\mathrm{opt}_1, \mathrm{opt}_2) = (\mathrm{pro}, \max)$. Now we have

$$\begin{aligned}\max\{1, er^{\mathrm{pro,max}}(\vec{x})\} \leq\ & \max\left\{1, \prod\nolimits_{i=1}^{n} \max\nolimits_{p\neq i} \frac{v_{ip}^*}{v_{ii}^*}\right\} \\ \leq\ & \max\left\{1, \alpha^{2m(n-1)} \prod\nolimits_{i=1}^{n} \max\nolimits_{p\neq i} \frac{v_{ip}}{v_{ii}}\right\} \\ \leq\ & \alpha^{2m(n-1)} \max\left\{1, er^{\mathrm{pro,max}}(\vec{v})\right\}.\end{aligned}$$

Choosing $\delta = n-1$ and applying inequality (4.4), we obtain (4.3).

Case 4: $(\mathrm{opt}_1, \mathrm{opt}_2) = (\max, \max)$. Finally, we have

$$\begin{aligned}\max\{1, er^{\mathrm{max,max}}(\vec{x})\} \leq\ & \max\left\{1, \max\nolimits_{i=1}^{n} \max\nolimits_{p\neq i} \frac{v_{ip}^*}{v_{ii}^*}\right\} \\ \leq\ & \max\left\{1, \alpha^{2m} \max\nolimits_{i=1}^{n} \max\nolimits_{p\neq i} \frac{v_{ip}}{v_{ii}}\right\} \\ \leq\ & \alpha^{2m} \max\left\{1, er^{\mathrm{max,max}}(\vec{v})\right\}.\end{aligned}$$

Choosing $\delta = 1$ and applying inequality (4.4), we again obtain (4.3).

Finally, it is again similar to the proof of Theorem 3.8, one can prove that the running time of Algorithm 6 is polynomial bounded by the input size and $1/\varepsilon$. This completes the proof. ❑

4.2.2 Inapproximability Result

In this section we prove that if the number of agents is part of the input, the problem Minimum Envy (max, max) does not have a polynomial-time approximation scheme (PTAS). This result can be extended without difficulty to the other cases of $(\text{opt}_1, \text{opt}_2)$ with $\text{opt}_i \in \{\max, \text{pro}\}$.

Theorem 4.2 *Unless* $\mathbf{P} = \mathbf{NP}$*, the problem* Minimum Envy $(\max, \max)$ *is not in* **PTAS**. *In fact, there is no approximation algorithm of factor better than* $3/2$ *for this problem, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. The proof is based on a reduction from the (X3C. Let $(\mathcal{B}, \mathcal{S})$, where $\|\mathcal{B}\| = 3q$ and $\mathcal{S} = \{S_1 \dots, S_n\}$, be an instance of X3C. Without of loss generality, we can assume that $n \geq q$. We construct an instance M as follows: There are n agents, each corresponding to one set S_i, $1 \leq i \leq n$; the set of goods contains $2q + n$ single goods, where $3q$ "real" goods correspond to the $3q$ elements of $\mathcal{B}$ and there are $n - q$ "dummy" goods. For each i, $1 \leq i \leq n$, agent a_i has utility 1 for each good in S_i and utility 0 for each good in $\mathcal{B} \smallsetminus S_i$. Every dummy good has utility 3 for all agents. We also denote by u_i the additive utility function of agent a_i.

Suppose that $(\mathcal{B}, \mathcal{S})$ is a yes-instance of X3C. Then there exists a set $I \subseteq \{1, \dots, n\}$, $\|I\| = q$, such that $S_i \cap S_j = \emptyset$ for all $i, j \in I$, $i \neq j$, and $\bigcup_{i \in I} S_i = \mathcal{B}$. Hence, we assign the bundle S_i to agent a_i for each $i \in I$, and each dummy good to one of the $n - q$ remaining agents. This allocation has an envy of 1 and thus is optimal.

Conversely, if $(\mathcal{B}, \mathcal{S})$ is a no-instance of X3C, we show that any optimal allocation for M has always envy of at least $3/2$. Indeed, let $\pi = (\pi_1, \dots \pi_n)$ be an optimal allocation for M, and consider the following two cases for π. First, if there is some agent a_i whose bundle π_i contains at least two dummy goods, then there must be another agent a_k owning a bundle π_k of value at most 3. This implies that

$$er^{\max,\max}(\pi) \geq \frac{u_k(\pi_i)}{u_k(\pi_k)} \geq 2$$

and thus the envy of π is at least 2 in this case. Second, consider the case that the $n-q$ dummy goods are assigned to $m-n$ distinct agents and let a_i be one of these. Since there are at most $q-1$ disjoint sets from $S_1, \ldots, S_n$, there must be at least one agent a_k who is assigned a bundle π_k of value at most 2. Furthermore, the bundle π_i of a_i has utility at least 3 for agent a_k. Hence,

$$er^{\max,\max}(\pi) \geq \frac{u_k(\pi_i)}{u_k(\pi_k)} \geq \frac{3}{2}$$

To sum up, the lower bound of $er^{\max,\max}(\pi)$ is $3/2$ if $(\mathcal{B}, \mathcal{S})$ is a no-instance of X3C. This means that an approximation algorithm with a factor better than $3/2$ will distinguish the yes- from the no-instances of X3C in polynomial time, contradicting **NP**-hardness of X3C unless $\mathbf{P} = \mathbf{NP}$. The proof is completed. ❑

4.2.3 A Restricted Case

We next consider the problem of minimizing the envy in the special case when there are as many agents as goods. By applying a matching technique, we will show that one can solve this restricted case in polynomial time. Like the problem of maximizing social welfare, it is important to note that each agent will get exactly one good, for otherwise there would be at least one agent owning nothing and thus the envy of the allocation would be infinite. Hence, one can transfer our optimization problem into a problem of finding a suitable maximum matching in a weighted bipartite graph. We consider the following variant of the matching problem MIN-MAX-MATCHING.

MIN-MAX-MATCHING	
Input:	A bipartite graph $G = (X \cup Y, E)$ and a weight function $w : E \to \mathbb{R}^+$.
Output:	A *min-max-matching*, i.e., a maximum matching $\mathcal{M} \subseteq E$ such that $\max_{e \in \mathcal{M}} w(e) \leq \max_{e \in \mathcal{M}'} w(e)$ for any other maximum matching $\mathcal{M}'$ of G.

Lemma 4.2 *Given a weighted bipartite graph G, a min-max-matching of G can be found in polynomial time.*

Proof. Given a weighted bipartite graph G, a min-max-matching of value k can be found in polynomial time as follows. Let G_k be a subgraph of G that contains only the edges of weight less than or equal to k. Obviously, G has a min-max-matching of value k if and only if G_k has a maximum matching. The smallest value of k can be found by binary search. ❑

Theorem 4.3 *For any pair* $(\text{opt}_1, \text{opt}_2)$ *with* $\text{opt}_i \in \{\max, \text{pro}\}$*, an allocation of minimum envy can be found in polynomial time if the number of agents and the number of goods are the same.*

Proof. Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a set of agents and $\mathcal{G} = \{g_1, \ldots, g_n\}$ be a set of goods, where we assume that each agent a_i has an additive utility function u_i over the set of goods. Consider the following two cases:

Case 1: $\text{opt}_1 = \max$. We construct a weighted bipartite graph $G = (X \cup Y, E)$ in which X and Y correspond to the set of agents $\mathcal{A}$ and the set of goods $\mathcal{G}$, respectively. The weight w function is defined as

$$w((a_i, g_j)) = \begin{cases} \max_{k \neq j} \left\{ \dfrac{u_i(g_k)}{u_i(g_j)} \right\} & \text{if} \quad \text{opt}_2 = \max \\ \prod_{k \neq j} \dfrac{u_i(g_k)}{u_i(g_j)} & \text{if} \quad \text{opt}_2 = \text{pro} \end{cases}$$

It is not difficult to see that the optimal allocation for instance M corresponds exactly to a min-max-matching $\mathcal{M}$ of G, which can be solved exactly by Lemma 4.2.

Case 2: $\text{opt}_1 = \text{pro}$. We construct a weighted bipartite graph $G = (X \cup Y, E)$ similarly as in the case 1, but this time the optimal allocation for instance M corresponds exactly to a MPMW $\mathcal{M}$ of G, which can be solved exactly by Lemma 3.1.

This completes the proof. ❑

Example 4.3 *Consider an instance of* Minimum Envy $(\max, \max)$ *with three agents* $\{a_1, a_2, a_3\}$ *and three goods* $\{r_1, r_2, r_3\}$*. The utility of each agent on each good is represented by a weighted complete bipartite graph as in Figure a (left). Figure b (right) illustrates the graph G equipped with the weight function w and the thick lines show an optimal allocation for M obtained by using the algorithm described in the proof of Lemma 4.2.*

Figure 4.1: An example for solving Minimum Envy (max, max) exactly when $m = n$.

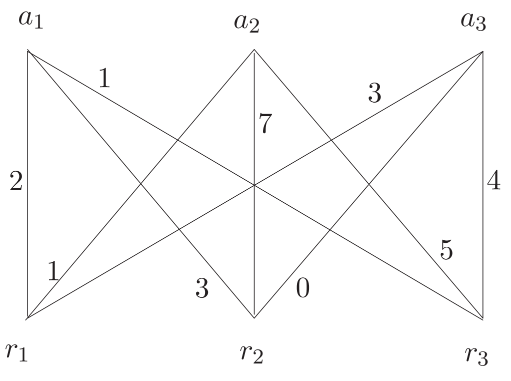


Figure a.

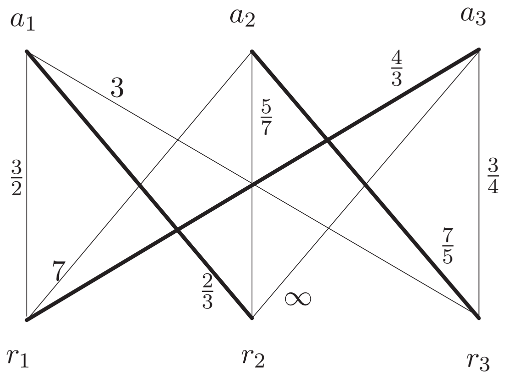


Figure b.

4.3 Conclusions and Future Work

We have studied the problem of computing minimum envy allocations of indivisible goods when agents have additive utilities over bundles of goods. Building on the work of Lipton et al. [LMMS04] and Chevaleyre et al. [CEEM07], we have analyzed an alternative metric to measure the envy between two agents and the envy of society as well. Based on these measures, we model the optimization problems of minimizing envy and study their approximability. Our main result shows that these problems admit an FPTAS for the case when the number of agents is not part of the input. We have also provided a hardness of constant-factor approximation result. For the restricted case when there are as many agents as goods, we have presented a nontrivial polynomial-time algorithm that computes exact allocations of minimum envy.

As future work, it would be interesting to find an (constant) approximation algorithm for our optimization problems in the general case. Note that our problem Minimum Envy (max, max) is closely related to the problem Minimum Makespan on unrelated machine which is known to be approximable to a factor of 2, by using integer linear programming (see [LST90]). Therefore, we conjecture that Minimum Envy (max, max) can be also approximated to within certain constant factor. Regarding the negative results, there is still room for improving the $3/2$-hardness factor obtained in Theorem 4.2.

Chapter 5

Truthful Mechanism Design

This chapter concerns with the inapproximabity results for truthful mechanisms for the problem of maximizing Nash social welfare and the problem of minimizing the envy which have been defined in the previous chapters.

5.1 Framework and Basic Definitions

In the framework of resource allocation, we usually make the assumption that the agents' utility functions are public information and efficient (exact or approximate) algorithms could be found based on this information. However, this convention cannot be applied well in certain situations where the agents' preferences or utilities over the resources are private and some agents might possibly misrepresent them in order to get better bundles. *Mechanism design* is a subfield of economics and game theory that answers the question of whether mechanisms can be designed that ensure that optimal allocations can be produced and all agents can get the maximal profit only by declaring their *truthful* utility functions.

Formally, Nisan and Ronen [NR99] model this problem as follows. Consider n *selfish* agents and m indivisible, nonshareable resources. Each agent a_i has a valuation function v_i and this information is known only to that agent herself.[1] The mechanism designer will require each agent a_i to report her valuation function and she may provide another function v'_i instead of her truthful one, v_i, in order to make the output of the algorithm beneficial for her. The algorithm will produce autonomously an optimal allocation based on the information provided by agents. Note that the mechanism design aims at computing an optimal solution with respect to the agents' true valuation functions, and

[1]Note that in this section the term *valuation function* plays the same role as "utility function" in the previous sections.

the basic idea is to give to each agent a *payment* function that gives her an incentive to report her true valuations.

Definition 5.1 *A* mechanism $\mathcal{M}$ *is a pair* $(f, \vec{p})$*, where*

- f *is an allocation algorithm,*
- $\vec{p} = (p_1, \dots, p_n)$ *is the vector of payment functions assigned to the agents.*

All the valuation functions provided by the agents are collected in a vector $\vec{v} = (v_1, \dots, v_n)$, and we denote by $f(\vec{v})$ the optimal allocation output by the algorithm. Each agent a_i is given a payment function $p_i(\vec{v})$ and she wishes to maximize her utility $u_i(\vec{v}) = v_i(f(\vec{v})) - p_i(\vec{v})$.

Definition 5.2 *A mechanism* $\mathcal{M}$ *is said to be* truthful *if for each agent* a_i *and for each possible misreported valuation* v'_i*,*

$$v_i(f(\vec{v})) - p_i(\vec{v}) \geq v_i(f(v'_i, \vec{v}_{-i})) - p_i(v'_i, \vec{v}_{-i}),$$

where $\vec{v}_{-i}$ *denotes the vector* $(v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n)$ *and* $(v'_i, \vec{v}_{-i})$ *denotes the vector* $(v_1, \dots, v_{i-1}, v'_i, v_{i+1}, \dots, v_n)$*.*

In the last few years, the study of truthful mechanisms has been one of the most interesting subjects, especially in the area of *combinatorial auctions.* Bikhchandani et al. [BCL+06] show that one can design efficient truthful mechanisms for the problem of maximum (utilitarian) social welfare in combinatorial auctions by a so-called VCG mechanism. Nisan and Ronen [NR99] investigated the design of computable truthful mechanisms for some combinatorial optimization problems including *shortest paths*, *minimum spanning trees*, and *minimum makespan.* Regarding negative results, Bikhchandani et al. [BCL+06] proved that there does not exist any truthful mechanism for the problem of maximum egalitarian social welfare, even in a restricted case, by taking into account the *weak monotonicity* property as a sufficient condition for the truthfulness of a mechanism.

Definition 5.3 *A truthful mechanism* $\mathcal{M}$ *is* weakly monotonic *if for each vector* $\vec{v}$*, agent* a_i*, and valuation function* v'_i*, we have*

$$v_i(f(\vec{v})) + v'_i(f(v'_i, \vec{v}_{-i})) \geq v'_i(f(\vec{v})) + v_i(f(v'_i, \vec{v}_{-i})).$$

For minimization problems, the inequality is reverse. The following lemma is due to Bikhchandani et al. [BCL$^+$06] and shows the relationship between truthfulness and weak monotonicity of a mechanism.

Lemma 5.1 (Bikhchandani et al. [BCL$^+$06]) *Any truthful mechanism must be weakly monotone.*

5.2 Our Results

Using Lemma 5.1, Mu'alem and Schapira [MS07] present lower bound results on truthfulness for a number of optimization problems. Specifically, they show that one cannot have a truthful mechanism that obtains any approximation to the problem of maximum egalitarian social welfare in resource allocation. In this section, we will study the question of whether one can design efficiently truthful mechanisms for MAX-TNSW, MAX-ANSW and MINIMUM ENVY $(\max, \max)$. The obtained results have been published in [NR13b, NR13a].

Theorem 5.1 *There does not exist any truthful mechanism that computes an optimal allocation for* MAX-TNSW *or for* MAX-ANSW.

Proof. We give the proof only for MAX-TNSW. Consider an instance of MAX-TNSW with two agents $\{a_1, a_2\}$ and two resources $\{r_1, r_2\}$. The valuation functions of the agents are additive and described as follows: $v_1(r_1) = 2$, $v_1(r_2) = 1$, $v_2(r_1) = 5$, $v_2(r_2) = 3$. For a contradiction, suppose that there is a truthful mechanism $\mathcal{M}$ computing an optimal allocation for MAX-TNSW. It is easy to see that $\mathcal{M}$ will return an allocation in which r_i is assigned to a_i, for $i \in \{1, 2\}$. Let $v_2'(r_1) = 2$, $v_2'(r_2) = 0.5$ be another valuation of agent a_2 over the resources. For this valuation function v_2', an optimal allocation obtained by $\mathcal{M}$ is that r_2 is assigned to a_1 and r_1 is assigned to a_2. We the have

$$3 + 2 = v_2(r_2) + v_2'(r_1) < v_2'(r_2) + v_2(r_1) = 0.5 + 5.$$

This contradicts weak monotonicity. The theorem is proved. ❑

Naturally, one would hope to design mechanisms that are truthful and achieve maximum Nash social welfare at least in an approximate manner. Unfortunately, this is also impossible due to the following result.

Theorem 5.2 *Let $0 < \varepsilon < 1$. There is no truthful mechanism that can yield an ε-approximation for* Max-TNSW *or for* Max-ANSW.

Proof. Again, we give the proof only for Max-TNSW. Consider an instance of Max-TNSW with two agents, $A = \{a_1, a_2\}$, and two resources, $R = \{r_1, r_2\}$. The valuation functions of the agents are additive and given as follows:

$$v_1(r_1) = 2,\ v_1(r_2) = \varepsilon,\ v_2(r_1) = 3 - \frac{\varepsilon}{2},\ v_2(r_2) = 1 + \frac{\varepsilon}{2}.$$

It is easily seen that there is an optimal allocation in which r_i is assigned to a_i, $i \in \{1, 2\}$. This allocation will be returned by any ε-approximation mechanism. Let $v_2'(r_1) = {}^{3\varepsilon}/_2$, $v_2'(r_2) = {}^{\varepsilon^2}/_2$ be another valuation of agent a_2 of the resources. For this valuation function u_2', an optimal allocation obtained by an ε-approximation mechanism is that a_1 gets r_2 and a_2 gets r_1. However, it is easy to check that the following is true for any $\varepsilon < 1$:

$$1 + 2\varepsilon = v_2(r_2) + v_2'(r_1) < v_2'(r_2) + v_2(r_1) = \frac{\varepsilon^2}{2} + 3 - \frac{\varepsilon}{2}.$$

This is again a contradiction to weak monotonicity. The proof is completed. ❑

Lipton et al. [LMMS04] proved that truthful mechanisms for finding exact solutions for the problem Minimum Envy $(\max, \max)$ are impossible. In the context of approximation algorithms, we will give a lower bound for this problem.

Theorem 5.3 *No truthful mechanism for* Minimum Envy $(\max, \max)$ *can have an approximation factor of $2 - \varepsilon$ for each fixed $\varepsilon > 0$.*

Proof. Let $\mathcal{M}$ be a truthful mechanism that achieves an approximation factor of $2 - \varepsilon$ for some $\varepsilon > 0$. We consider an instance with two agents, $A = \{a_1, a_2\}$, and three resources, $R = \{r_1, r_2, r_3\}$. All agents have the same additive valuation functions: $v_i(r_j) = 1$ for $i \in \{1, 2\}$ and $j \in \{1, 2, 3\}$. It is easy to see that one can obtain an optimal allocation with a value of 2 by assigning two resources to one agent and the remaining resource to the other agent. Furthermore, $\mathcal{M}$ will also output a solution with a value of 2 and, w.l.o.g., we may assume that the bundle $\{r_1, r_2\}$ is assigned to a_1 and the single resource r_3 belongs to a_2.

Let $v_1'(r_1) = v_1'(r_2) = {}^1/_2, v_1'(r_3) = {}^3/_2$ be another valuation function of agent a_1 over the resources. Note that the new instance has an optimal allocation with a value of 1 when assigning resources r_1 and r_2 to agent a_2 and resource r_3 to agent a_1, whereas all other allocations have a value of at least 2. Hence, the $(2-\varepsilon)$-approximation mechanism returns exactly the optimal allocation. We then have

$$2 + 3/2 = v_1(r_1, r_2) + v'_1(r_3) > v'_1(r_1, r_2) + v_1(r_3) = 1 + 1.$$

This is again a contradiction to weak monotonicity. ❑

5.3 Conclusion and Future Work

In this chapter, we have studied and provided the lower bound results for (deterministic) truthful mechanisms of three problems: Max-TNSW, Max-ANSW and Minimum Envy $(\max, \max)$. For the first two problems, we have shown that there is no truthful mechanism for finding both exact and approximate solutions. In addition, we have presented a lower bound of 2 on the approximation factor of truthful mechanisms for Minimum Envy $(\max, \max)$. An obvious direction of future work is to design a truthful mechanisms that can approximate this problem within a certain factor. It would also be interesting to study *truthful randomized mechanism* (that is, a probability distribution over deterministic truthful mechanisms) for both Max-TNSW and Max-ANSW.

Chapter 6

Positional Scoring Rules for Multiagent Resource Allocation

Most of the work so far has been focused on allocating indivisible goods that involves cardinal preferences of agents over alternative bundles of goods. These preferences are represented typically in term of utility functions which map alternatives to a suitable scale. Admittedly, the use of utility functions seems to be technically convenient in computational practice. However, in many realistic settings eliciting exact utilities of agents over the bundle of goods that they received is not always easy, especially in scenarios where monetary payments are not allowed. This chapter is concerned with the ordinal aspect of preferences where agents express their preferences through binary relations instead of utility functions.

Division of indivisible goods based on the ordinal preference has been studied in the sequence of papers [BF00, EF01, HP02, BEF04, BK05, BEL10, BKK12]. Herreiner and Puppe [HP02] design a deterministic procedure, called *descending demand procedure*, for allocating indivisible goods. The difficulty of their approach, however, lies in the assumption that agents should be able to express the linear order preferences over all bundles of goods, which, in the worst case, requires agents to express an exponentially large input, which must be avoided for computational reasons. Indeed, with only 30 goods, agents would have to rank over more than one billion different bundles of goods. Bourveret, Endriss and Lang [BEL10] study a model where agents' preferences are expressed in term of the so-called *SCI-nets*. Intuitively, given a linear order $\succ$ over single goods, a SCI-net induces an incomplete preference order over bundles which is consistent with $\succ$. In particular, Bourveret and coauthors propose the refinements of envy-freeness, called *possibly envy-free* and *necessarily envy-free*, which are a synthesis of similar notions introduced by Brams et al. [BEF04]. Among other results, they show that possibly envy-free allocations are easy to compute while determining necessarily envy-free allocations are **NP**-hard even when the number of goods is as twice as large as the the number of agents. Brams, Edelman, and Fishburn [BEF04] considered scenario where agents rank individual goods only and have additively separable preferences. Based on Borda scoring vector, they defined two types of allocation: *maxsum* (utilitarianism) and *maxmin* (egalitarianism). Here the Borda score of object r_i for agent a_j ranges from 1 (when r_i is a_j's least preferred object) to m (when r_i is a_j's most preferred object), and the score total of an agent is the sum of the Borda scores of the objects assigned to her. While the maxsum allocations aim at maximizing the sum of Borda scores of all agents, the maximin allocations seek to maximize the minimum score of any agent.

It is motivated by the work of Brams *et al.* [BEF04], we will generalize their Borda-optimal allocations [BEF04] to arbitrary scoring vectors and aggregation functions, and study them in detail, from the point of view of social choice and computational complexity. Beyond Borda, the scoring vectors we consider are k-approval (each agent attaches a score of 1 to her k most preferred objects and zero to the other objects), lexicographicity (a k^{th} ranked good counts more than the sum of all objects that have the lower ranks), and quasi-indifference (all objects have roughly the same score, up to very small differences). As for aggregation functions, we focus on utilitarianism ($\star = +$), egalitarianism ($\star = \min$), $\star =$ leximin, (which in a *strict sense* is not an aggregation function), as well as $\star =$ envy, which is an useful measure for minimizing the difference between score of agents.

6.1 Positional Scoring Allocation Rules

Typically, an allocation rule for dividing m indivisible goods among agents is characterized by a scoring vector and a social welfare aggregation function. Similarly to positional scoring voting rules in voting, a scoring vector is defined as a vector $s = (s_1, \dots, s_m) \in \mathbb{R}^m$ satisfying $s_1 \geq \cdots \geq s_m \geq 0$, where for each agent's ranking, s_1 is the score of her top-ranked good, s_2 is the score of her second-ranked good, and so on. The final score of an allocation for an agent is the sum of the scores of the goods assigned to her. The aggregation function $\star$ (+, min, leximin and envy) aggregate the individual scores of all agents toward an optimal allocation. We now describe the model of our problems in detail.

Let $A = \{a_1, \ldots, a_n\}$ be a set of agents and $R = \{r_1, \ldots, r_m\}$ a set of indivisible goods. An *allocation* is a partition $\pi = (\pi_1, \ldots, \pi_n)$, where $\pi_i \subseteq R$ is the bundle of goods assigned to agent a_i. Since the number of all subsets of goods are exponential in m a linear order preference over 2^R is impossible. Hence, we assume that each agent a_i provides her strict ranking $\succ_i$ on the set of single goods only. Under this assumption, a *profile* $P = (\succ_1, \ldots, \succ_n)$ is a collection of n rankings over R, and a *allocation rule* (respectively, a *allocation correspondence*) maps any profile to an allocation (respectively, a nonempty subset of allocations).

We now define a family of allocation rules that more or less corresponds to the family of positional scoring rules in voting (see, e.g., [BF02]).

Definition 6.1 *A* scoring vector *is a vector $s = (s_1, \ldots, s_m)$ of real numbers such that $s_1 \geq \cdots \geq s_m \geq 0$ and $s_1 > 0$. Given a preference ranking $\succ$ over R and $r \in R$, let $rank(r, \succ)$ denote the* rank of r under $\succ$. *The* utility function over 2^R induced by the ranking $\succ$ on R and the scoring vector s *is for each bundle $B \subseteq R$ defined by*

$$u_{\succ,s}(B) = \sum_{r \in B} s_{rank(r,\succ)}.$$

We consider the following specific scoring vectors:

- Borda scoring: borda $= (m, m-1, \ldots, 1)$,[1]
- lexicographic scoring: lex $= (2^{m-1}, 2^{m-2}, \ldots, 1)$,
- quasi-indifference for some ε, $0 < \varepsilon \ll 1$:
  ε-qi $= (1 + (m-1)\varepsilon, 1 + (m-2)\varepsilon, \ldots, 1)$.
- k-approval: k-app $= (1, \ldots, 1, 0, \ldots, 0)$, where the first k entries are ones and all remaining entries are zero.

Example 6.1 *Let $R = \{a, b, c\}$ be a set of three goods and let two agents have the following preference profile: $(a \succ b \succ c,\ b \succ c \succ a)$. Let $\pi = (\{a\}, \{b, c\})$. Then, for the Borda scoring vector, agent 1's bundle $\{a\}$ has value 3 and agent 2's bundle $\{b, c\}$ has value $3 + 2 = 5$.*

[1]Note that the usual definition of the Borda scoring vector in voting is $(m-1, m-2, \ldots, 1, 0)$. Here, together with [BEF04] we fix the score of the bottom-rank object to 1, meaning that getting it is better than nothing. For scoring voting rules, a translation of the scoring vector has obviously no impact on the winner(s); for allocation rules, however, it does.

The individual utilities are then aggregated using a monotonic, symmetric *aggregation function* that is then to be optimized. The four we will use here are among the most obvious ones: *sum* (utilitarianism), *min*, *leximin* (two versions of egalitarianism), and *envy* (that minimizes the envy of an allocation). Leximin refers to the (strict) lexicographic preorder over utility vectors whose components have been preordered nondecreasingly. Formally, for $x = (x_1, \ldots, x_n)$, let $x' = (x'_1, \ldots, x'_n)$ denote some vector that results from x by rearranging the components of x nondecreasingly, and define $x <_{\text{leximin}} y$ if and only if there is some i, $0 \leq i < n$, such that $x'_j = y'_j$ for all j, $1 \leq j \leq i$, and $x'_{i+1} < y'_{i+1}$.

For each scoring vector s, we thus have four *allocation correspondences*:

- $F_{s,+}(P) = \text{argmax}_\pi \sum_{1 \leq i \leq n} u_{\succ_i, s}(\pi_i)$,
- $F_{s,\min}(P) = \text{argmax}_\pi \min_{1 \leq i \leq n} \{u_{\succ_i, s}(\pi_i)\}$, and
- $F_{s,\text{leximin}}(P) = \text{argmax}_\pi \text{leximin}(u_{\succ_1, s}(\pi_1), \ldots, u_{\succ_n, s}(\pi_n))$,
- $F_{s,\text{envy}}(P) = \text{argmin}_\pi \max_{j \neq i} \left\{1, \frac{u_{\succ_i, s}(\pi_j)}{u_{\succ_i, s}(\pi_i)}\right\}$,

where $P = (\succ_1, \ldots, \succ_n)$ is a profile and $\pi = (\pi_1, \ldots, \pi_n)$ an allocation. Whenever we write $F_{s,\star}$, we mean any one of $F_{s,+}$, $F_{s,\min}$, $F_{s,\text{leximin}}$, and $F_{s,\text{envy}}$.

Example 6.2 *For $n = 2$ agents and $m = 5$ goods, $R = \{a, b, c, d, e\}$, assume we have the profile $P = (abcde, bcdea)$ (where the leftmost item is the most preferred one). The optimal allocations are described in Table 6.1. For the case of the 3-approval, the single good e can be assigned to any agent without changing the social welfare. In case of the lexicographic scoring vector, the envy of an allocation is minimized if and only if a is assigned to agent 1 and b is assigned to agent 2; other remaining goods c, d, e can be assigned arbitrary to agents.*

Table 6.1: Optimal allocations for the instance given in Example 6.2.

Agg. function	Borda	quasi-ind	lex	3-approval
$+$	$(a, bcde)$	$(a, bcde)$	$(a, bcde)$	$(a, bcd), (abc, d)$ $(ab, cd), (ac, bd)$
min	(ab, cde)	(ab, cde)	(ac, bde)	$(ab, cd), (ac, bd)$
leximin	(ab, cde)	(ab, cde)	(ac, bde)	$(ab, cd), (ac, bd)$
envy	$(ab, cde), (ac, bde)$	$(ab, cde), (ac, bde)$	$(a?, b?)$	$(ab, cd), (ac, bd)$

6.2 Problem Modeling

For a given scoring vector s and a given aggregation function $F_{s,\text{envy}}$, we define the following decision problem associated with the value of an optimal allocation.

$F_{s,\text{envy}}$-OPTIMAL-ALLOCATION-VALUE ($F_{s,\text{envy}}$-OAV)	
Given:	A profile $P = (\succ_1, \dots, \succ_n)$ of n agents' rankings on a set R of indivisible goods and a nonnegative integer t.
Question:	Is there an allocation $\pi = (\pi_1, \dots, \pi_n)$ with $\max_{j \neq i} \left\{1, \frac{u_{\succ_i,s}(\pi_j)}{u_{\succ_i,s}(\pi_i)}\right\} \leq t$?

Analogously, we define $F_{s,+}$-OAV by asking whether $\sum_{1 \leq i \leq n} u_{\succ_i,s}(\pi_i) \geq t$, and $F_{s,\min}$-OAV whether $\min_{1 \leq i \leq n} u_{\succ_i,s}(\pi_i) \geq t$, and $F_{s,\text{leximin}}$-OAV where the bound is an ordered list $(t_1, \dots, t_n)$ of nonnegative integers and we ask whether $(u_{\succ_1,s}(\pi_1), \dots, u_{\succ_n,s}(\pi_n)) >_{\text{leximin}} (t_1, \dots, t_n)$.

Furthermore, we also consider the corresponding optimization version of the decision problem above.

$F_{s,\text{envy}}$-FIND-OPTIMAL-ALLOCATION ($F_{s,\text{envy}}$-FOA)	
Input:	A profile $P = (\succ_1, \dots, \succ_n)$ of n agents' rankings on a set R of indivisible goods.
Output:	An allocation $\pi = (\pi_1, \dots, \pi_n)$ that minimizes $\max_{i \neq j} \left\{1, \frac{u_{\succ_i,s}(\pi_j)}{u_{\succ_i,s}(\pi_i)}\right\}$.

The problems $F_{s,+}$-FOA, $F_{s,\min}$-FOA and $F_{s,\text{leximin}}$-FOA can be defined similarly. It is easy to see that $F_{s,+}$-FOA (and thus $F_{s,+}$-OAV) are solvable in polynomial time for any scoring vector s. Nevertheless, $F_{s,\star}$-FOA and $F_{s,\star}$-OAV are much more complicated for any $\star \in \{\min, \text{leximin}, \text{envy}\}$. For the case of the k-approval scoring vector, $F_{k\text{-app},\min}$-FOA and $F_{k\text{-app},\min}$-OAV are restricted cases of the problem of maximizing egalitarian social welfare with $\{0,1\}$-additive functions, which can be solved in polynomial time by a network flow algorithm (see Golovin [Gol05]).

6.3 Our Results

In this section, we focus on the complexity of "winner determination" for a few key combinations of a scoring vector and an aggregation function, considering both decision and functional problems. In addition, we give several approximation results, one of which makes use of a so-called *picking sequences.*

6.3.1 Complexity of Winner Determination

The complexity of determining an optimal allocation for a given scoring vector and a given aggregation function has been established by Baumeister et al. [BBL$^+$]. Namely, they showed that $F_{\epsilon\text{-qi,min}}$-OAV, $F_{\epsilon\text{-qi,leximin}}$-OAV, $F_{\text{lex,min}}$-OAV and $F_{\text{lex,leximin}}$-OAV are **NP**-complete by using the reduction from X3C. We now extend these results to the problems $F_{\epsilon\text{-qi,envy}}$-OAV and $F_{\text{lex,envy}}$-OAV.

Theorem 6.1 $F_{\epsilon\text{-qi,envy}}$-OAV *is* **NP**-*complete.*

Proof. Recall that an instance of X3C includes a finite set $\mathcal{B}$ with $\|\mathcal{B}\| = 3q$ and a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of 3-element subsets of $\mathcal{B}$. The question is whether there is a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that every element of $\mathcal{B}$ occurs in exactly one member of $\mathcal{S}'$? Let $(\mathcal{B}, \mathcal{S})$ be an instance of X3C (w.l.o.g assuming $n > q$), we construct an instance of $F_{\epsilon\text{-qi,envy}}$-OAV as follows. There are n agents $\{a_1, \ldots, a_n\}$, each agent a_i corresponds to a set S_i, $1 \le i \le n$. We create a set $R = \mathcal{B} \cup \mathcal{D}$ of $m = 4n - q$ goods, where $\mathcal{B}$ is a set of $3q$ "real" goods, each corresponds to an element of $\mathcal{B}$, and $\mathcal{D}$ is a set of $4(n-q)$ "dummy" goods. Agent a_i has the following preferences: $S_i \succ_i \mathcal{B} \setminus S_i \succ_i \mathcal{D}$, where a set $\mathcal{T}$ in this order stands for all the goods of $\mathcal{T}$ in any fixed order. We now prove that $(\mathcal{B}, \mathcal{S})$ is a yes-instance of X3C if and only if there exists an allocation of goods to agents such that it has the value of at most

$$t = \frac{4 + (8n - 8q - 2)\varepsilon}{3 + (12n - 3q - 6)\varepsilon}$$

($\Rightarrow$) Suppose that $(\mathcal{B}, \mathcal{S})$ is a yes-instance of X3C and let $\mathcal{S}' \subseteq \mathcal{S}$ be an exact cover of $\mathcal{B}$. An allocation π whose value is exactly t can be computed as follows. For every $S_i \in \mathcal{S}'$, assign a bundle corresponding to S_i, each of value $3+(12p-3q-6)\varepsilon$, to agent a_i. Next, we assign the dummy goods to $n-q$ remaining agents in a way such that they get the bundles of the same value. Without loss of generality, we assume that $\mathcal{D} = \{r_1, r_2, \ldots, r_{4(n-q)}\}$ and $r_1 \succ r_2 \succ \cdots \succ r_{4(n-q)}$, then we can divide $\mathcal{D}$ into $n - q$ disjoint subsets, each of the form $\{r_j, r_{j+1}, r_{4(n-q)-j}, r_{4(n-q)-j+1}\}$, $j = 1, 3, \ldots, 2(n-q) - 1$. It is easy to see that each of such subsets has the value of $4 + (8p - 8q - 2)\varepsilon$. Hence, the value of the allocation π equals to t.

($\Leftarrow$) Assume that $(\mathcal{B}, \mathcal{S})$ is a no-instance of X3C and let π be an optimal allocation for $F_{\epsilon\text{-qi,envy}}$-OAV. By the definition of quasi-indifference scoring vector, there will be q agents, each receives three goods from $\mathcal{B}$, and other $n-q$ agents, each gets four dummy goods of the same value from $\mathcal{D}$. Since there is no exact cover of $\mathcal{B}$ there must be at least one agent who does not get the first three goods at her top-rank. This means the value of π is greater than t. Hence, the instance constructed from $(\mathcal{B}, \mathcal{S})$ is no-instance of $F_{\epsilon\text{-qi,envy}}$-OAV. ❑

Theorem 6.2 $F_{\text{lex,envy}}$-OAV *is* **NP**-*complete.*

Proof. The proof is again by a reduction from X3C. Given an instance $(\mathcal{B}, \mathcal{S})$ with $\mathcal{S} = \{S_1, \ldots, S_n\}$ and $\|\mathcal{B}\| = 3q$, an instance of $F_{\text{lex,envy}}$-OAV is constructed with n agents and $n+2q$ goods. The set of goods is $R = \mathcal{B} \cup \mathcal{D}$, where $\mathcal{B}$ is set of $3q$ "real" goods, each corresponds to an element of $\mathcal{B}$, and $\mathcal{D}$ is a set of "dummy" goods but at this time $\|\mathcal{D}\| = n - q$. Agent a_i has the following preferences: $\mathcal{D} >_i S_i >_i \mathcal{B} \setminus S_i$. One can prove that $(\mathcal{B}, \mathcal{S})$ is a yes-instance of X3C if and only if there exists an allocation for the new constructed instance of $F_{\text{lex,envy}}$-OAV with the value of at most

$$t = \frac{2^{2q+n}}{2^{3q-1} + 2^{3q-2} + 2^{3q-3}}$$

($\Rightarrow$) Suppose that $(\mathcal{B}, \mathcal{S})$ is a yes-instance of X3C and let $\mathcal{S}'$ be an exact cover of $\mathcal{B}$. We compute an allocation π as follows. For each i such that $S_i \in \mathcal{S}$, we assign the bundle S_i to the corresponding agent a_i. Each of $n-q$ remaining agents will get a dummy good from $\mathcal{D}$. It is easy to see that the value of the allocation π is exactly t.

($\Leftarrow$) Assume that there is no exact cover for $(\mathcal{B}, \mathcal{S})$. Let π be an optimal allocation for the instance that has been constructed from $(\mathcal{B}, \mathcal{S})$. Since $\|\mathcal{D}\| = n - q$, there will be q agents who do not get any dummy goods from $\mathcal{D}$. Moreover, at least one of them, say a_i, cannot receive the bundle S_i. Hence, for this agent, her utility is of at most $2^{3q-1} + 2^{3q-2} + 2^{3q-3} - 1$. This implies that the value of the allocation π is greater than t. ❑

For a special case when the number of agents is not the part of input, we provide efficient algorithms for both decision and optimization problems by using dynamic programming.

Theorem 6.3 *For any scoring vector $s \in \{\text{borda}, \epsilon\text{-qi}, \text{lex}\}$ and for any aggregation function $\star \in \{\min, \text{leximin}, \text{envy}\}$, the problems $F_{s,\star}$-OAV and $F_{s,\star}$-FOA are solvable in polynomial time whenever the number of agents is constant.*

Proof. It is enough to prove the theorem for $F_{s,\text{leximin}}$-FOA.

To show that the $F_{s,\text{leximin}}$-FOA is solvable in polynomial time for Borda and quasi-indifference scoring vector, we consider the following algorithm. Let $R = \{r_1, \dots, r_m\}$ be a set of m goods and $A = \{a_1, \dots, a_n\}$ be a set of n agents. We compute the individual utility of every agent for all possible allocations that assign the first j goods to the n agents, encoded as an n-dimensional vector. Let V_0 be the set containing only the vector $\vec{0}$. In each step j, for each vector $\vec{v} \in V_{j-1}$ compute one vector $v_i = \vec{v} + s_{rank(r_j, \succ_i)} \cdot \vec{e_i}$ for each agent a_i, $1 \leq i \leq n$, where $\vec{e_i}$ denotes the i-th unit vector. The vector v_i is added to V_j. It is easy to see that $\|V_j\| \leq \|V_m\|$ for all $j \leq m$.

For $s = \text{borda}$, every entry of each vector in V_m is bounded above by $^{m(m+1)}/_2$ and thus $\|V_m\| \in O(m^{2n})$. For $s = \epsilon$-qi, every entry of each vector in V_m has the form $p + q \cdot \varepsilon$, where $p, q \in \mathbb{Z}$, $0 \leq p \leq m$ and $0 \leq q \leq {}^{m(m-1)}/_2$. Hence, $\|V_m\| \in O(m^{3n})$. It is not difficult to see that the running time of the algorithm depends on $\|V_m\|$ and thus is polynomial in m.

We now turn to lexicographic scoring, $s = \text{lex}$. The case with two agents can be solved efficiently by implementing the following simple rules when the preferences of the agents are examined from the most to the least preferred good:

- **Case 1:** If the agents have different goods that are not assigned yet on the current position, both agents get their current goods and proceed with the next position.
- **Case 2:** If both current objects are already assigned, proceed with the next position.
- **Case 3:** If both agents rank the same object, say r, that is not assigned yet on the current position, then let $r_i \neq r$ be the most preferred good of agent $a_i, i \in \{1, 2\}$ that has not been assigned yet, and w.l.o.g. assume that $rank(r_1, \succ_1) \geq rank(r_2, \succ_2)$. Assign r to agent a_1 and all remaining objects to agent a_2.
- **Case 4:** The last case is that only one of the current objects, say r (w.l.o.g., the one ranked by agent a_1), has not been assigned yet. If r is not the most preferred good of agent a_2 among those not yet assigned, then assign it to agent a_1 and remaining objects to agent a_2. Otherwise, let $r_i \neq r$ be the most preferred

good of agent $a_i, i \in \{1, 2\}$ that has not been assigned yet. Note that $rank(r, \succ_1) < rank(r, \succ_2)$. (i) If $rank(r_1, \succ_1) < rank(r, \succ_2)$, agent 1 receives r_1, while agent 2 gets r and the remaining objects. (ii) If $rank(r_1, \succ_1) = rank(r, \succ_2)$, assign r_1 to agent a_1 and g to agent 2, and proceed with the next position. (iii) If $rank(r, \succ_2) < rank(r_1, \succ_1) < rank(r_2, \succ_2)$, assign r to agent a_2 and the remaining objects to agent a_1. (iv) If $rank(r_1, \succ_1) \geq rank(r_2, \succ_2)$, give r to agent a_1 and the remaining objects to agent a_2.

Obviously, the running time of the algorithm above is polynomial in m.

Example 6.3 *Consider an instance with two agents $\{a_1, a_2\}$ and six single goods $\{a, b, c, d, e, f\}$. The preferences of agents are given in Table 6.2, where the goods are arranged from the top to the bottom according to their rank. The goods that are in bold face show an optimal allocation.*

Table 6.2: Optimal allocations for the problem with two agents, lexicographic scoring vector and min (leximin) aggregate function.

Ranking							Case 4							
of	Case 1		Case 2		Case 3		(i)		(ii)		(iii)		(iv)	
goods	a_1	a_2	a_1	a_2	a_1	a_2	a_1	a_2	a_1	a_2	a_1	a_2	a_1	a_2
1	$\boldsymbol{a}$	a	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{a}$	$\boldsymbol{b}$
2	b	$\boldsymbol{c}$	$\boldsymbol{d}$	$\boldsymbol{e}$	$\boldsymbol{c}$	a	$\boldsymbol{e}$	$\boldsymbol{f}$	c	a	$\boldsymbol{e}$	$\boldsymbol{f}$	$\boldsymbol{e}$	$\boldsymbol{f}$
3	c	$\boldsymbol{d}$	$\boldsymbol{c}$	c	b	$\boldsymbol{d}$	c	a	$\boldsymbol{d}$	$\boldsymbol{c}$	c	a	$\boldsymbol{c}$	a
4	d	$\boldsymbol{e}$	b	$\boldsymbol{f}$	d	$\boldsymbol{e}$	$\boldsymbol{d}$	e	b	d	b	$\boldsymbol{c}$	d	$\boldsymbol{d}$
5	e	$\boldsymbol{b}$	e	a	e	c	b	$\boldsymbol{c}$	$\boldsymbol{f}$	$\boldsymbol{e}$	$\boldsymbol{d}$	e	b	c
6	f	$\boldsymbol{f}$	f	d	f	$\boldsymbol{f}$	f	d	e	f	f	d	f	e

We now prove by induction on the (constant) number n that the problem $F_{s,\text{leximin}}$-FOA is solvable in polynomial time, and the number of optimal solutions is always bounded by a constant. Indeed, the induction base $n = 2$ has been already shown. Also, by the algorithm described above, there are at most two optimal allocations for any instance with two agents. Assume now that the claim is true for $n-1$, where $n > 2$. Consider an instance I with a set A of n agents and and a set R of m goods and a profile $P = (\succ_1, \ldots, \succ_n)$ of agents' ranking over the single goods. It is not difficult to show that every optimal allocation for I must have the egalitarian utility of at least 2^{m-n}, that is, each agent must be received at least one good from her n most preferred goods.[2] Formally, if $\pi = (\pi_1, \ldots, \pi_n)$ is an optimal allocation then $\pi_i \cap S_i \neq \emptyset$, where S_i denotes the set of the first n goods of agent a_i's ranking, for all i, $1 \leq i \leq n$. Furthermore, there is at least one agent who receives her top-rank good. We denote by D the subset of $S = S_1 \times \cdots \times S_n$ such that for any vector $\mu = (r_1^\mu, \ldots, r_n^\mu) \in D$, $r_i^\mu \neq r_j^\mu$ for all $i \neq j$, and there is at least one good r_i^μ that is the most preferred object of agent a_i, for some i. Obviously, $\|D\| \leq \|S\| \in \mathcal{O}(1)$ since n is constant. We next define the set

$$W = \text{argmax}_\mu \min_{1 \leq i \leq n} 2^{m-rank(r_i^\mu, \succ_i)}, \ \mu \in D$$

It is easy to see that the set W contains a constant number of vectors and can be computed in $\mathcal{O}(1)$ time. For each vector $w = (r_1^w, \ldots, r_n^w) \in W$, let:

$$k^w = \min_{1 \leq i \leq n} \{rank(r_i^w, \succ_i)\}$$

and

$$A^w = \{a_i \in A |\, rank(r_i^w, \succ_i) = k^w\} \text{ and } R^w = R \setminus \{r_1^w, \ldots, r_n^w\}$$

Let $\pi^w = (\pi_j^w)_{a_j \in A^w}$ denote an optimal allocation for the new instance with $\|A^w\|$ agents and $\|R^w\|$ goods. One can prove that such an optimal allocation π^w can be computed in polynomial time. Indeed, let us consider two cases below:

- $k = 1$: this means there are not any two agents having the same good at top of their ranking. For this case we assign n top-rank goods to n agents and remove them from the set R. So, now instead of solving I, we only need to solve the new instance I' with n agents and $m - n$ goods.

- $k \geq 2$: for each vector $w \in W$, there is at least one agent receiving her most preferred object and she will not be in A^w. Hence, we have $\|A^w\| \leq \|A\| - 1$. By the induction hypothesis, the set V of all optimal allocations π^w of (A^w, R^w) can be found in polynomial time and $\|V\|$ is bounded by a constant.

[2] Note that we need only consider the case $m > n$, otherwise the problem can be solved in $\mathcal{O}(1)$ time since n, m is constant.

A possible optimal allocation $\mu^w = (\mu_1^w, \dots, \mu_n^w)$ for I will be a "combination" between the vector $w \in W$ and a vector $\pi^w \in V$. Formally, μ^w is determined as follows:

$$\mu_i^w = \begin{cases} \{r_i^w\} & \text{if } a_i \notin A^w, \\ \{r_i^w\} \cup \{\pi_i^w\} & \text{otherwise} \end{cases} \qquad (i = 1, \dots, n)$$

Since the size of both W and V is bounded by a constant the number of such allocations μ^w is also bounded by a constant. Hence, an optimal allocation for the original instance I can be found by exhaustive search in time $\mathcal{O}(1)$. This completes the proof. ❑

Example 6.4 *Consider an instance I with 5 agents and 10 goods, the ranking of agents over single goods are given in the Table 6.3.*

Table 6.3: The ranking of agents over the single goods for the instance given in Example 6.4.

Ranking	Agent a_1	Agent a_2	Agent a_3	Agent a_4	Agent a_5
1	r_1	r_2	r_1	r_3	r_2
2	r_2	r_3	r_4	r_5	r_5
3	r_3	r_8	r_8	r_7	r_3
4	r_4	r_{10}	r_9	r_{10}	r_9
5	r_9	r_6	r_{10}	r_2	r_7
6	r_6	r_5	r_5	r_4	r_{10}
7	r_7	r_9	r_6	r_6	r_8
8	r_5	r_7	r_3	r_8	r_6
9	r_{10}	r_4	r_7	r_1	r_1
10	r_8	r_1	r_2	r_9	r_4

The set W contains two vectors:

$$w_1 = (r_1, r_2, r_4, r_3, r_5), w_2 = (r_1, r_3, r_4, r_5, r_2)$$

- Consider $w_1 = (r_1, r_2, r_4, r_3, r_5)$, we have:

$$A^{w_1} = \{a_3, a_5\} \qquad \textit{and} \qquad R^{w_1} = \{r_6, r_7, r_8, r_9, r_{10}\}$$

An unique optimal allocation for the instance (A^{w_1}, R^{w_1}) with two agents is

$$\pi^{w_1} = (\{r_8\}, \{r_6, r_7, r_9, r_{10}\})$$

Therefore, a possible optimal allocation for the original instance I is

$$\mu^{w_1} = (r_1, r_2, \{r_4, r_8\}, r_3, \{r_5, r_6, r_7, r_9, r_{10}\})$$

- Consider $w_2 = (r_1, r_3, r_4, r_5, r_2)$, we have:

$$A^{w_2} = \{a_2, a_3, a_4\} \qquad \text{and} \qquad R^{w_2} = \{r_6, r_7, r_8, r_9, r_{10}\}$$

For the new instance (A^{w_2}, R^{w_2}), there are two optimal allocations:

$$\pi^{w_2} = (r_8, \{r_6, r_9, r_{10}\}, r_7)$$
$$\pi'^{w_2} = (\{r_6, r_9, r_{10}\}, r_8, r_7)$$

Hence, the possible optimal allocations for the original instance I are:

$$\mu^{w_2} = (r_1, \{r_3, r_8\}, \{r_4, r_6, r_9, r_{10}\}, \{r_5, r_7\}, r_2)$$
$$\mu'^{w_2} = (r_1, \{r_3, r_6, r_9, r_{10}\}, \{r_4, r_8\}, \{r_5, r_7\}, r_2)$$

By comparing between μ^{w_1}, μ^{w_2} and μ'^{w_2}, we find that μ^{w_1} is the unique optimal allocation for I.

6.3.2 Approximation

This section deals with the approximability of the problem $F_{s,\star}$-FOA for a scoring vectors $s \in \{\text{lex}, \epsilon\text{-qi}\}$ and for an aggregation functions $\star \in \{\min, \text{envy}\}$. We first give a simple approximation algorithm within a factor of 1/2 for $F_{\text{lex},\min}$-FOA. The proof is based on the matching technique.

Theorem 6.4 *There exists a (1/2)-approximation algorithm for $F_{\text{lex},\min}$-FOA.*

Proof. Given an instance with a set A of n agents, a set R of m goods and a profile $P = (\succ_1, \dots, \succ_n)$, our algorithm is described as follows. We first construct a weighted complete bipartite graph $G = (A \times R, E)$: a vertex a_i for every agent i, a vertex r_j for every good r_j, and a weight function w which is defined as $w(a_i, r_j) = s_{rank(r_j, \succ_i)}$. We create a graph G' by deleting all the edges in G of the weight less than 2^{m-1}. If there exits a maximum matching $\mathcal{M}$ of G' then return $\mathcal{M}$. Otherwise, we replace $m-1$ by $m-2$ and repeat until we find a maximum matching.

A matching $\mathcal{M}$ returned by the algorithm will correspond to an incomplete allocation of m goods to n agents. By assigning all remaining goods to agents greedily, we obtain a complete allocation δ. We now prove that δ has the egalitarian social welfare within a factor of $1/2$ of the optimum. Indeed, assume that π is an optimal allocation. The collective utility of π has the form of $(2^{k_1}+\alpha_1,\ldots,2^{k_n}+\alpha_n)$ where $m-n \le k_i \le m-1$ and $\alpha_i < 2^{k_i}$. Obviously, an allocation that has the collective utility of $(2^{k_1},\ldots,2^{k_n})$ will be exactly corresponding to a maximum matching of G. Without loss of generality, we can assume that $k_n = \min\{k_1,\ldots,k_n\}$. Then it is easy to see that the minimum weight edge in $\mathcal{M}$ must have the weight of 2^{k_n}. Finally, the egalitarian social welfare of δ is at least of

$$2^{k_n} > \frac{1}{2}(2^{k_n}+\alpha_n) = \min\{2^{k_1}+\alpha_1,\ldots,2^{k_n}+\alpha_n\}$$

The proof is completed. ❑

While the hardness results for $F_{\text{borda,min}}$-FOA and $F_{\text{borda,envy}}$-FOA are still not known, Baumeister et al. [BBL$^+$] studied the approximability of $F_{\text{borda,min}}$-FOA using two simple protocols *regular picking sequences* $(1\ldots,n)^*$ and *fair picking sequences* $(1\ldots nn\ldots 1)^*$. For these protocols, at each step, a designated agent will pick his most preferred object among those have not been assigned yet. One may wonder why these protocols deserve to be investigated. The reasons are twofold: one, picking sequences are very cheap in communication, because agents only reveal part of their preferences by picking objects, and two, they are very easy to implement and have linear running time in m, where m is number of objects.

We next show that when the number of goods is twice as large as the number of agents, the fair picking sequence yields a $1/2$-approximation algorithm for $F_{\text{borda,envy}}$-FOA.

Theorem 6.5 *For $m = kn$ objects, fair picking sequence is 2-approximation algorithm for $F_{\text{borda,envy}}$-FOA.*

Proof. Let $\pi = (\pi_1,\ldots,\pi_n)$ be an allocation obtained by applying the fair picking sequence. We will prove that for every agent a_i:

$$\frac{u_i(\pi_j)}{u_i(\pi_i)} \le 2 - \frac{2}{n} - \frac{n-2}{n(m+1)} \tag{6.1}$$

for all $j \neq i$. In fact, because of the symmetry, we need only to prove the claim for the case $i = 1$, the other cases can be proved similarly.

We consider two cases as follows:

Case 1: $m = 2kn$. According to the picking policy, agent a_1 will pick an object at steps $1, 2n, 2n+1, 4n, 4n+1 \ldots, 2(k-1)n+1, 2kn$. Hence, in the worst case, agent a_1 gets objects ranked in the positions $2(j-1)n+1, 2jn$, for $j = 1, \ldots, k$ and thus the score of the bundle she received is:

$$\begin{aligned}
u_1(\pi_1) &\geq 1 + 2n + (2n+1) + 4n + \cdots + [2(k-1)n+1] + 2kn \\
&= k + 2n \cdot [1 + 3 + \cdots + (2k-1)] \\
&= k + 2nk^2 \\
&= \frac{m}{2n} + 2n\frac{m^2}{4n^2} \\
&= \frac{m(m+1)}{2n}.
\end{aligned}$$

The evaluation of agent i's bundle, $i \neq 1$, by agent a_1 is maximal if they have the same ranking. In this case, agent a_i gets the objects she and agent a_1 ranked $2, 3, 6, 7, \ldots, m-2, m-1$. Hence, we have:

$$\begin{aligned}
u_1(\pi_i) &\leq m - 1 + m - 2 + m - 5 + m - 6 + \cdots + m - 4k + 3 + m - 4k + 2 \\
&= k(2m - 4k + 1) \\
&= \frac{m}{2n}\left(m - \frac{m}{2n} + 1\right).
\end{aligned}$$

It follows that:

$$\begin{aligned}
a = \frac{u_1(\pi_i)}{u_1(\pi_1)} = \frac{\frac{m}{2n}\left(m - \frac{m}{2n} + 1\right)}{\frac{m(m+1)}{2n}} &= \frac{2mn - 2m + n}{n(m+1)} \\
&= 2 - \frac{2}{n} - \frac{n-2}{n(m+1)}.
\end{aligned}$$

Case 2: $m = (2k+1)n$. By the same argument for the case 1, we can compute the score of agent a_1 in the worst-off case is

$$u_1(\pi_1) \geq \frac{1}{n}\sum_{j=1}^{2kn}(n+j) + m - 2kn = m + k + 2k^2 n,$$

and for any bundle π_i of agent a_i, $i \neq 1$:

$$u_1(\pi_i) \leq k(2m - 4k + 1) + m - 4k - 1.$$

Hence, we have:

$$b = \frac{u_1(\pi_i)}{u_1(\pi_1)} = \frac{k(2m - 4k + 1) + m - 4k - 1}{m + k + 2k^2 n}.$$

Replacing $k = \frac{m-n}{2n}$ and simplification, we obtain:

$$\begin{aligned} b &= \frac{2mn(m-n) - 2(m-n)^2 + 2mn^2 - 3n(m-n) - 2n^2}{2mn^2 + n(m-n) + n(m-n)^2} \\ &= \frac{2m^2 n + mn - n^2 - 2m^2}{n(n^2 + m - n + m^2)}. \end{aligned}$$

Finally, we prove that $b < a$. Indeed,

$$\begin{aligned} b < a &\iff \frac{2m^2 n + mn - n^2 - 2m^2}{n(n^2 + m - n + m^2)} < \frac{2mn - 2m + n}{n(m+1)} \\ &\iff 2mn^3 - 4mn^2 + 2mn + n^3 > 0 \\ &\iff 2m(n-1)^2 + n^2 > 0. \end{aligned}$$

The last inequality holds for all $m, n > 0$, thus $b < a$. ❑

6.4 Conclusion and Future Work

We have studied the positional scoring rules for multiagent allocation of indivisible goods. We have generalized the results of Brams et al. [BEF04] for variety of aggregation functions and for several types of scoring vector. Tables 6.4 and 6.5 summarize the obtained results in this chapter. These tables also mention some open questions that would be interesting for the future research. Among those, the most important one could be finding the missing complexity results for the Borda case.

Table 6.4: Overview of complexity results with respect to positional scoring rules.

	OAV	FOA	Reference
$F_{s,+}$	in **P**	**PO**	
$F_{s,\text{min}}$	**NP**-comp*	**NP**-hard*	
lex or ε-qi	**NP**-comp	**NP**-hard	[BBL+]
borda	open	open	
lex or borda or ε-qi, if $n \in O(1)$	**P**	**PO**	Theorem 6.3
$F_{s,\text{leximin}}$	**NP**-comp*	**NP**-hard*	
lex or ε-qi	**NP**-comp	**NP**-hard	[BBL+]
borda	open	open	
lex or borda or ε-qi, if $n \in O(1)$	**P**	**PO**	Theorem 6.3
$F_{s,\text{envy}}$	**NP**-comp*	**NP**-hard*	
lex or ε-qi	**NP**-comp	**NP**-hard	Theorem 6.1 & 6.2
borda	open	open	
lex or borda or ε-qi, if $n \in O(1)$	**P**	**PO**	Theorem 6.3

*if s is part of the input (even for two agents with same preferences)

Table 6.5: Overview of approximability results with respect to positional scoring rules.

Scoring rules	min	Reference	envy	Reference
Borda ($m = kn$)	1/2	[BBL+]	2	Theorem 6.5
lex	1/2	Theorem 6.4	open	

Bibliography

[AAWY97] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 493–500. Society for Industrial and Applied Mathematics, January 1997.

[AAWY98] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

[ABSS93] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optimia in lattices, codes, and systems of linear equations. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 724–733, 1993.

[AC04] L. Ausubel and P. Cramton. Auctioning many divisible goods. *Journal of the European Economic Association*, 2(2-3):480–493, 2004.

[ACG$^+$99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties.* Springer, 1999.

[ACG05] J. Abrache, T. Crainic, and M. Gendereau. Models for bundle trading in financial markets. *European Journal of Operation Research*, 160:88–105, 2005.

[AFS08] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. In *Proceedings of Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques. 11th International Workshop APPROX 2008 and 12th International Workshop RANDOM 2008*, pages 10–20. Springer-Verlag *Lecture Notes in Computer Science #5171*, August 2008.

[AFS12] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24, 2012.

[AKS02] M. Agrawal, N. Kayal, and N. Saxena. Primes is in **P**. *Annals of Mathematics*, 2:781–793, 2002.

[AL96] 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.

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[Aro94] S. Arora. *Probabilistic checking of proofs and the hardness of approximation problems*. PhD thesis, UC Berkeley, 1994.

[Aro98] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of **NP**. *Journal of the ACM*, 45(1):70–122, 1998.

[AS07] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 114–121. ACM Press, July 2007.

[AS10] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM Journal on Computing*, 39(7):2970–2989, 2010.

[BBL$^+$] D. Baumeister, S. Bouveret, J. Lang, T. Nguyen, J. Rothe, and A. Saffidine. Positional scoring rules for the allocation of indivisible goods. Submited.

[BCL$^+$06] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[BD05] I. Bezáková and V. Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005.

[BE05] R. Beigel and D. Eppstein. 3-coloring in time $\mathcal{O}(1.3289^n)$. *Journal of Algorithms*, 54(2):168–204, 2005.

[BEF04] S. Brams, P. Edelman, and P. Fishburn. Fair division of indivisible items. *Theory and Decision*, 5(2):147–180, 2004.

[BEL10] S. Bouveret, U. Endriss, and J. Lang. Fair division under ordinal preferences: Computing envy-free allocations of indivisible goods. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 387–392. IOS Press, 2010.

[BF00] S. Brams and P. Fishburn. Fair divisionof indivisible items between two people with identical preferences: Envy-freeness, pareto-optimality. *Social Choice and Welfare*, 17(2):247–267, 2000.

[BF02] S. Brams and P. Fishburn. Voting procedures. In K. Arrow, A. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare*, volume 1, pages 173–236. North-Holland, 2002.

[BH92] R. Boppana and M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, 1992.

[BH06] A. Björklund and T. Husfeldt. Inclusion–exclusion algorithms for counting set partitions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 575–582, 2006.

[BK05] S. Brams and D. King. Efficient fair division-help the worst off or avoid envy. *Rationality and Society*, 17(4):387–421, 2005.

[BKK12] S. Brams, D. Kilgour, and C. Klamler. The undercut procedure: an algorithm for the envy-free division of indivisible items. *Social Choice and Welfare*, 39(2-3):615–631, 2012.

[BL08] S. Bouveret and J. Lang. Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. *Journal of Artificial Intelligence Research*, 32:525–564, 2008.

[BLFL05] S. Bouveret, M. Lemaître, H. Fargier, and J. Lang. Allocation of indivisible goods: a general model and some complexity results. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1309–1310, 2005.

[BN07] L. Blumrosen and N. Nisan. Combinatorial auctions. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 11, pages 267–299. Cambridge University Press, 2007.

[BS06] N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 31–40. ACM Press, July 2006.

[BST98] C. Bazgan, M. Santha, and Z. Tuza. Efficient approximation algorithms for the subset-sums equality problem. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, pages 387–396. Springer-Verlag *Lecture Notes in Computer Science #1443*, 1998.

[BT95] S. Brams and A. Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

[BT96] S. Brams and A. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution.* Cambridge University Press, 1996.

[CCK09] D. Chakrabarty, J. Chuzhoy, and S. Khanna. On allocating goods to maximize fairness. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 107–116. IEEE Computer Society Press, 2009.

[CCPV07] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract).

In *Proceedings of the 12th International Integer Programming and Combinatorial Optimization Conference*, pages 182–196. Springer-Verlag *Lecture Notes in Computer Science #4513*, June 2007.

[CDE⁺06] 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.

[CEEM04] 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.

[CEEM07] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Reaching envy-free states in distributed negotiation settings. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1239–1244. IJCAI, January 2007.

[CEEM08] 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.

[CKW92] J. Csirik, H. Kellerer, and G. Woeginger. The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters*, 11(5):281–287, 1992.

[CLRS09] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[CSS05] 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.

[CSS06] P. Cramton, Y. Shoham, and R. Steinberg. *Combinatorial Auctions*. MIT Press, 2006.

[CVZ10] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, pages 575–584. IEEE Computer Society, 2010.

[DFL82] B. Deuermeyer, D. Friesen, and M. Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic and Discrete Methods*, 3(2):452–454, 1982.

[Din07] I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[DNS10] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. *Mathematics of Operations Research*, 35(1):1–13, 2010.

[DW11] D. Shmoys D. Williamson. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.

[DWL05] P. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. *Artificial Intelligence*, 164(1–2):23–46, 2005.

[EF01] P. Edelman and P. Fishburn. Fair division of indivisible items among people with similar preferences. *Mathematical Social Sciences*, 41(3):327–347, 2001.

[Eso01] M. Eso. An iterative auction for online seats. In *Mathematics of the Internet: E-Auctions and Markets.* Springer-Verlag, Berlin, 2001.

[Fei08] U. Feige. On allocations that maximize fairness. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 287–293. Society for Industrial and Applied Mathematics, January 2008.

[FGMS06] L. Fleischer, M. Goemans, V. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 611–620. ACM Press, 2006.

[FK10] F. Fomin and D. Kratsch. *Exact exponential algorithms.* Springer, 2010.

[GA99] G. Gambosi V. Kann A. Marchetti-Spaccamela M. Protasi G. Ausiello, P. Crescenzi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer Verlag, 1999.

[GHIM09] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 535–544. Society for Industrial and Applied Mathematics, January 2009.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[GMR97] M. Golfarelli, D. Maio, and A. Rizzi. A task swap negotiation protocol based on the contract net paradigm. Technical report, CSITE, University Bologna, 1997.

[Gol05] D. Golovin. Max-min fair allocation of indivisible goods. Technical Report CMU-CS-05-144, School of Computer Science, Carnegie Mellon University, June 2005.

[Gra66] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[Gra69] R. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

[Gra97] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92(2):167–189, 1997.

[Hås99] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[Hås01] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[Hoc95] D. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, 1995.

[HP02] D. Herreiner and C. Puppe. A simple procedure for finding equitable allocations of indivisible goods. *Social Choice and Welfare*, 19(2):415–430, 2002.

[Hro01] J. Hromkovič. *Algorithmics for hard problems-Introduction to combinatorial optimization, randomization, approximation, and heuristics.* Springer, 2001.

[Hro05] J. Hromkovič. *Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms.* Springer, 2005.

[HS74] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21(2):277–292, 1974.

[HS76] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2):317–327, 1976.

[HS87] D. Hochbaum and D. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.

[IK75] O. Ibarra and C. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

[Joh50] N. John. The bargaining problem. *Econometrica*, 18(2):155–162, April 1950.

[Joh74] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[Kar72] R. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[KGMO07] S. Khot, G., E. Mossel, and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

[Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 767–775, 2002.

[KK98] M. Kovalyov and W. Kubiak. A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs. *Journal of Heuristics*, 3(4):287–297, 1998.

[KLMM08] S. Khot, R. Lipton, E. Markakis, and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52(1):3–18, 2008.

[KP07] S. Khot and A. Ponnuswami. Approximation algorithms for the max-min allocation problem. In *Proceedings of Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques. 10th International Workshop APPROX 2007 and 11th International Workshop RANDOM 2007*, pages 204–217. Springer-Verlag *Lecture Notes in Computer Science #4627*, 2007.

[KPW94] M. Kovalyov, C. Potts, and L. Wassenhove. A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work. *Mathematics of Operations Research*, 19(1):86–93, 1994.

[Law76] E. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5(3):66–67, 1976.

[Len83] H. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[LLN01] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pages 18–28. ACM Press, 2001.

[LMMS04] 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.

[LOS99] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 96–102. ACM Press, 1999.

[LOS02] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[LST90] J. Lenstra, D. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1):259–271, 1990.

[Mac94] J. MacMillan. Spelling spectrum rights. *Journal of Economic perspective*, 8:145–162, 1994.

[MN99] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[Mou88] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[Mou04] H. Moulin. *Fair Division and Collective Welfare*. MIT Press, 2004.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MS85] B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22(1):115–123, 1985.

[MS07] A. Mu'alem and M. Schapira. Setting lower bounds on truthfulness (extended abstract). In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, pages 1143–1152. Society for Industrial and Applied Mathematics, January 2007.

[MU05] M. Mitzenmacher and E. Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[NNRR] N. Nguyen, T. Nguyen, M. Roos, and J. Rothe. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Journal of Autonomous Agents and Multiagent Systems*. To appear.

[NNRR12a] N. Nguyen, T. Nguyen, M. Roos, and J. Rothe. Complexity and approximability of egalitarian and Nash product social welfare optimization in multiagent resource allocation. In *Proceedings of the 6th European Starting AI Researcher Symposium*, pages 204–215. IOS Press, August 2012.

[NNRR12b] N. Nguyen, T. Nguyen, M. Roos, and J. Rothe. Complexity and approximability of social welfare optimization in multiagent resource allocation (extended abstract). In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems,*, pages 1287–1288. IFAAMAS, June 2012.

[NNRR12c] N. Nguyen, T. Nguyen, M. Roos, and J. Rothe. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. In *Proceedings of the 4th International Workshop on Computational Social Choice*, pages 335–346, 2012.

[NR] T. Nguyen and J. Rothe. How to decrease the degree of envy in allocations of indivisible goods. In *Proceedings of the Third International Conference on Algorithmic Decision Theory*. To appear.

[NR99] N. Nisan and A. Ronen. Algorithmic mechanism design (extended abstract). In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 129–140. ACM Press, May 1999.

[NR13a] T. Nguyen and J. Rothe. Envy-ratio and average-Nash social welfare optimization in multiagent resource allocation. In *Proceedings of the 6th International Workshop on Optimisation in Multi-Agent Systems*, pages 1–18, May 2013.

[NR13b] T. Nguyen and J. Rothe. Envy-ratio and average-Nash social welfare optimization in multiagent resource allocation (extended abstract). In *Proceedings of the 12th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1139–1140. IFAAMAS, May 2013.

[NRR] T. Nguyen, M. Roos, and J. Rothe. A survey of approximability and inapproximability results for social welfare optimization in multiagent resource allocation. *Annals of Mathematics and Artificial Intelligence*. To appear.

[NRR12] 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*, 2012.

[Pap95] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, second edition, 1995.

[Pol48] G. Polya. *How to solve it.* Princeton University Press, 1948.

[PW92] C. Potts and L. Wassenhove. Approximation algorithms for scheduling a single machine to minimize total late work. *Operations Research Letters*, 11(5):261–353, 1992.

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[Rob01] J. Robson. Finding a maximum independent set in time $\mathcal{O}(2^{n/4})$. Technical report, LaBRI, Université Bordeaux I, 2001.

[Rot64] G. Rota. On the foundations of combinatorial theory I. Theory of Möbius functions. *Probability Theory and Related Fields*, 2(4):340–368, 1964.

[Rot05] J. Rothe. *Complexity Theory and Cryptology. An Introduction to Cryptocomplexity.* EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2005.

[RP85] J. Romanycia and J. Pelletier. What is heuristic? *Computational Intelligence*, 1(2):47–58, 1985.

[RPH98] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[RR10] 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.

[RRSY07] T. Riege, J. Rothe, H. Spakowski, and M. Yamamoto. An improved exact algorithm for the domatic number problem. *Information Processing Letters*, 101(3):101–106, 2007.

[RSB82] S. Rassenti, V. Smith, and R. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13(2):402–417, 1982.

[RW98] J. Robertson and W. Webb. *Cake-cutting algorithms: Be fair if you can.* AK Peters, 1998.

[Sah76] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.

[San93] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 256–262, 1993.

[Sch03] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, 2003.

[Sch05] U. Schöning. Algorithmics in exponential time. In *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science*, pages 36–43, 2005.

[Sch08] D. Scheder. Guided search and a faster deterministic algorithm for 3-SAT. In *Proceedings of the 8th Latin American Symposium*, pages 60–71, 2008.

[SG76] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[Sil04] A. Silver. An overview of heuristic solution methods. *Journal of the Operational Research Society*, 55(9):936–956, 2004.

[Smi80] R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29:1104–1113, 1980.

[SRN03] P. Sousa, C. Ramos, and J. Neves. The fabricare scheduling prototype suite: Agent interaction and knowledge base. *Journal of Intelligent Manufacturing*, 14(3):441–455, 2003.

[TT77] R. Tarjan and A. Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

[Vaz03] V. Vazirani. *Approximation Algorithms.* Springer-Verlag, second edition, 2003.

[Von08] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, pages 67–74. ACM Press, July 2008.

[Woe97] G. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997.

[Woe01] G. Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization*, pages 185–208, 2001.