Automated Mechanism Design
A Large Scale Optimization Approach

Kia Babashahi Ashtiani

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

May 2019

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By:	Kia Babashahi Ashtiani
Entitled:	Automated Mechanism Design a Large scale Optimization approach

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair

______________________________ Examiner
Dr. Tiberiu Popa

______________________________ Examiner
Dr. Yann-Gaël Guéhéneuc

______________________________ Supervisor
Dr. Brigitte Jaumard

Approved ______________________________
Chair of Department or Graduate Program Director

__________ 20 ____ ______________________________
Dr. Lata Narayanan, Dean
Faculty of Engineering and Computer Science

Abstract

Automated Mechanism Design
A Large Scale Optimization Approach

Kia Babashahi Ashtiani

A set of self-interested and rational agents in a social network want to distribute their initial set of resources among themselves to obtain the set of resources they desire the most. Each agent, being self-interested and rational has incentive to lie about its preferences towards the resources to manipulate the outcome of the system to its advantage. In the context of multi-agent systems, Automated Mechanism Design (AMD), is a computer based design of rules that allows the reach of an equilibrium despite the selfishness of its agents.

Most of the multi-agent and AMD research has focused on one-to-one bilateral exchanges that only allow two agents to exchange resources.

Very few studies have been conducted on multilateral (many-to-many) types of trade. While several multi-agent algorithms exist for the one-to-one case, very few algorithms exists for the many-to-many case, which is more often encountered in social networks. AMD algorithms lack scalability as they rely on the enumeration of the resource allocation combinations.

We first propose three new optimization models for the AMD problem using a decomposition technique to create mechanisms that are not only scalable, i.e., one could use them to solve the AMD problem for data sets consisting of hundreds of agents and resources in seconds, but also support different types of trades between different numbers of agents (one-to-one, many-to-one and many-to-many). Then we illustrate a new mathematical model with a polynomial number of variables corresponding to the compact formulation of the current model of the literature that supports the many-to-many type of trade.

Numerical experiments show that we can solve significantly larger data sets than the ones existing in the literature, i.e., **up to 2,000 agents and 2,000 resources for the many-to-many type of trade in less than 24 seconds.**

First, I would like to thank god who helped me obtain this degree and was on my side through some of the most difficult times of my life.

Second, I would like thank my family who supported me with their unconditional love. I could not have asked for a better family and truth is, where ever I am now and whoever I will become in the future is thanks to you and all the great things that you did for me. Thanks mom, thanks dad, thanks Kasra.

Last but not least, I would also like to express my gratitude, love and respect towards my wonderful supervisor, professor Brigitte Jaumard for all she has taught me, all her support, patience and kindness. Thank you so much professor, you are the best.

The ink of the scholar is more sacred than the blood of the martyr.

Muhammad

Contents

Chapter 1

Introduction

1.1 The Multi-agent Resource Allocation Problem

The multi-agent resource allocation problem is concerned with allocating a set of resources among a set of agents. It can be solved via either a centralized approach or a distributed one. In a centralized approach, agents report their preferences towards the resources to a central entity and the central entity determines the final resource allocation between the agents. On the contrary to a centralized approach, in a distributed approach, agents exchange their set of resources with each other by the means of local negotiations. The negotiation process continue until either an equilibrium is reached or a certain time limit has elapsed.

1.2 Mechanism Design

Mechanism Design is a sub-field of Economics and Game Theory concerned with the design of the rules of interactions among a set of self-interested and rational agents who want to maximize their utility (Parkes [2001]), i.e., their measure of happiness. The multi-agent resource allocation problem can be modeled as a mechanism design problem when the agents are self-interested and rational with each having some private information regarding their preferences towards the resources which could result in the agents having incentive to lie about their preferences towards the resources to manipulate the outcome of the mechanism to their advantage.

1.3 Automated Mechanism Design Problem

In this thesis, we follow Sandholm's work in (Sandholm [2003]). He introduced a new approach towards solving the Mechanism Design problem called Automated Mechanism Design

(AMD) by viewing the Mechanism Design problem as an optimization problem and showing that optimization algorithms can be used to solve it.

To have an optimization formulation, all of the properties of the mechanism have to be explicitly translated into constraints of the optimization model. It is important to make sure that properties such as the rationality of the agents, the types of the resources, the utility functions, the selected negotiation protocol(s) and objective function(s) are explicitly translated into the optimization model as each one of these components could impact the final outcome of the mechanism.

The AMD problem can be solved for either divisible or indivisible resources. A resource is called divisible if it can be broken down into smaller pieces and still be of value where as an indivisible resource is only accepted as a whole. In this thesis we assume the resources to be indivisible. Agents could use different types of utility functions to evaluate their preference towards the resources. For example, they can use modular (additive), sub-modular or super-modular utility functions. In this thesis, we assumed that all agents have additive or modular utility functions. Meaning that each agent's utility towards a resource bundle is the summation over the agents utility for each of the elements in the bundle.

One can solve the AMD problem using different objective functions. Some might want to increase the utility of the agent with the smallest initial utility, i.e., the poorest agent, whereas some might want to give advantage to the richest one. Some others might just want to be fair. Money can or can not be allowed in the system. In this thesis, we consider the maximization of the utilitarian social welfare for the mechanism and assume no money is allowed.

Different negotiation protocols can also be considered. One could set limitations on the types of trades either with respect to the number of resources or the number agents that are involved in a trade. So far in the literature, mostly bilateral trades have been considered as the multilateral trades face immediate scalability issues. In this thesis, we tackle different types of trades involving two agents, i.e., one-to-one trades/bi-lateral, or multiple agents and multiple resources, i.e., many-to-many/multilateral trades.

1.4 Contributions

In his work, Sandholm (T.Sandholm [1998]) discussed the effect of different negotiation protocols/trades on reaching an optimal resource allocation and the computational complexity of the system. Furthermore, most of the focus of the literature has been on bilateral trades for which only two agents can negotiate their set of resources amongst themselves. As the AMD problem will face scalability issues with an increase in the number of agents or resources.

We first start with the formulation introduced by Sandholm in (Sandholm [2003]) which consists of an exponential number of variables and re-interpret it as a decomposotion model which can be solved via the Column Generation Algorithm (e.g., (Chvatal [1983])). Indeed, Column Generation is a tool used in large scale optimization which has proven to be extremely efficient for solving problems dealing with an exponential number of variables: rather than considering a mathematical program with all the variables/columns, it defines a restricted mathematical program with a small explicit number of variables/columns, and consider the other ones implicitly only.

Later, by taking advantage of the structure of the sub problem associated with the decomposed model, i.e., the so-called pricing problem in Operations Research, we will design different variations of the problem corresponding to different types of trades. Specifically, we will design three different variations of our column generation model, namely the one-to-many Column Generation model, the one-to-one Column Generation model and then the many-to-many column generation model (CG_AMD), which are much more scalable than the models existing in the literature and correspond to different types of trades.

Lastly, we introduce a fourth model, called the COMPACT_AMD, for the many-to-many types of trade with a polynomial number of variables. We next show that it corresponds to the compact formulation of the many-to-many column generation model (CG_AMD). The COMPACT_AMD surpasses its corresponding column generation model (CG_AMD) in terms of computational time and is more scalable, i.e., few seconds for instances with up to 2,000 agents or 2,000 resources.

The thesis relies on two papers on which we worked during my M.Comp.Sci. studies. We discuss the literature review in Chapter 2. Chapter 3 contains our first paper for which we presented a column generation model for many-to-many trades and the compact formulation associated with it. The second paper is presented in Chapter 4. It contains different models corresponding to different types of trades and a comparison among them in terms of their objective value and computational time. Conclusions and future work will be discussed in Chapter 5.

Chapter 2

Literature review

In the context of the multi-agent resource allocation problem, finding an efficient allocation of resources is not an easy quest as current solution schemes have difficulties to scale (Chevaleyre *et al.* [2006]). The multi-agent resource allocation problem has been approached by different communities of scientists from Computer Scientists to Economists due to its many applications.

In 1998, (T.Sandholm [1998]) proposed a formal description of the multi-agent resource allocation problem by analyzing the effect of limiting the types of trades among the agents, with respect to the social welfare of the multi-agent system and the time it requires to reach an optimal allocation of resources. Sandholm started by discussing bilateral transactions in which only two agents were allowed to exchange resources among themselves and eventually discussed multilateral transactions in which any agent could trade any number of resources in his possession with any other agent in his neighbourhood. Moreover, Sandholm showed that reaching the optimal resource allocation is a necessary condition for allowing multilateral transactions among the agents.

The multi-agent resource allocation problem can be solved via a centralized or a distributed approach. In a centralized approach, a central entity decides on the final resource allocation of the system. This entity might not be trustworthy or might have limited computational power (Chevaleyre et al. [2017]). On the contrary to a centralized approach, in a distributed approach the resource allocation evolves by the means of local negotiations between the agents, hence there is no need for a central entity.

Even in a distributed framework, the multi-agent resource allocation problem remains difficult to solve. Most of the existing work in the literature is concerned with bilateral trades, e.g., (Park and Yang [2007]). A reason for this choice could be that solving the multi-agent resource allocation problem by allowing multilateral transactions, which is much more common in practice, faces scalability issues as the search space of the problem is greater than in the centralized case. Hence, authors approached the multi-agent resource allocation problem from different perspectives and set some set of restrictions on the types of the trades (Chevaleyre et al. [2007]). Different types of limitations can be considered: they can be either applied to the number of agents involved in a transaction or to the number of resources.

(Conitzer and Sandholm [2002]) approached the resource allocation problem from a more economical perspective by introducing the mechanism design problem. The problem is concerned with a set of self-interested and rational agents and a set of resources with each agent having some private information regarding its preference towards the resources. Later, (Sandholm [2003]) modeled the Mechanism Design (MD) problem as an optimization model and used linear programming and optimization tools to solve it.

Authors have discussed the Mechanism Design problem by trying to solve it for different objectives. For example (Chevaleyre et al. [2007]) emphasised the fairness and envy-freeness of the mechanism and tried to design a mechanism to reach an envy free and fair distribution of resources. They have also tried to solve the problem by defining payment functions and using them to compensate the amount of loss of an agent to stay as fair as possible.

Other authors have analyzed the effect of some other properties on the outcome of the mechanism like the divisibility of the resources or different negotiation protocols. For example, (Endriss et al. [2006]) have analyzed different negotiation frameworks and trades with respect to divisible or indivisible resources.

Finally, different types of utility functions and their effects on the final resource allocation and the social welfare as well as their effect on the fairness of the mechanism have also been analyzed. Indeed having different utility functions such as sub-modular, modular (additive) or super-modular could affect the outcome of the mechanism as discussed in (Chevaleyre et al. [2017]).

Continuing the work which has been done in the literature, in this thesis, we tackled the Automated Mechanism Design problem by considering the scalability issues, the optimal resource allocation and having different types of trades in the mechanism. We used some large scale optimization techniques and algorithms to design different mechanisms corresponding to different types of trades among the agents. Having applied the algorithms and the models in this thesis we could solve the AMD problem for much larger data sets than the ones existing in the literature regardless of when having many-to-many, one-to-many or one-to-one trades.

Chapter 3

Optimal Automated Mechanism Design via two Linear Programming Models

The following chapter is based on the first paper that we have worked on during my M.Comp.Sci. degree. It is currently submitted for publication to an international reviewed conference.

In the sequel, we will solve the AMD problem for the most common type of trade in the industry, namely the many-to-many/multi-lateral type of trade. Following what was done previously in (Asselin et al. [2006]), we will re-interpret the model proposed by (Sandholm [2003]) as a decomposed model which can be solved via a column generation algorithm and then illustrate an new linear program with a polynomial number of variables(COMPACT_AMD), corresponding to compact formulation of the decomposed model and conclude the paper by comparing the two models in terms of their CPU time.

3.1 Abstract

In the context of multi-agent systems, Automated Mechanism Design (AMD) is the computer-based design of the rules of a mechanism, which reaches an equilibrium despite the fact that agents may be selfish and lie about their preferences. Although it has already been shown that AMD can be modelled as a linear program, there is no known efficient algorithm because of the exponential number of variables for such a model.

We revisit the first linear program model proposed for the AMD problem and introduce a new one with a polynomial number of variables. After having shown that the first model corresponds to a Dantzig-Wolfe decomposition of the second, we design efficient solution schemes in polynomial time for both models.

Numerical experiments compare the efficiency of both models and show that we can solve much larger data instances than before, up to 2,000 agents or 2,000 resources in about 35 seconds.

3.2 Introduction

The multi-agent resource allocation problem is at the interface of Computer Science and Economics and has been studied for a long time, either with a centralized or a distributed allocation framework. In a centralized approach, the agents report their preferences on the resources to the auctioneer, which determines the final resource allocation. In a distributed approach, the initial resource allocation evolves by local negotiations among the agents. Being guaranteed to end the negotiation process on an optimal or a near-optimal solution remains a critical issue as determining the most promising transactions among more than two agents is difficult, but plays a central role toward the negotiation process.

The efficient allocation of resources is complex and its automated mathematical solution quickly shows scalability issues (Chevaleyre *et al.* [2006]). A formal classification of the transactions has been proposed in (T.Sandholm [1998]): It starts with the one-to-one transactions, which only allow for one task to move from one agent to another at a time. It then goes on with the multi-agent contract and the combination of all of them, i.e., either one-to-many transactions (one agent initiator can negotiate simultaneously with several other agents), or many-to-many transactions (agents can negotiate simultaneously with any other agent). Sandholm (T.Sandholm [1998]) has shown that the multilateral transactions are essential in order to guarantee the reachability of an optimal resource allocation. Very few studies have been carried out on many-to-many transactions due to the difficulties of designing them (Dash et al. [2003]), and therefore of implementing them with scalable algorithms.

Some studies (Andersson and Sandholm [1998]) suggest tackling the scalability issue by using a restricted set of transactions; complex transactions can only be used when the simplest ones are no longer useful. However, this iterated process only leads, most of the time, to a local optimum. (Chevaleyre et al. [2008]) show that it is easy (e.g., simple negotiation protocols are sufficient) to reach a socially optimal allocation of resources when one can use qualitative measures, with agent preference relations over alternative bundles of resources. Generalizations are possible with k-additive utility functions where the agent utility of a resource bundle R is defined by the summation of basic utilities ascribed to subsets of R with cardinality $\leq k$. The impact of parameters, such as restrictions on the transaction types or on the agent behaviors, has been also studied (Chevaleyre et al. [2007]). However, simulations are, most of the time, restricted to bilateral transactions and with no information on the distance of the output solution to an optimal solution. A recent survey by (Ye et al. [2017]) discuss both the theoretical challenges and the various applications of self-organized multi-agent systems.

3.2.1 Distributed Multi-agent Systems

The multi-agent resource allocation problem is defined by a set of autonomous agents and an initial resource allocation. The system evolves through local negotiations and the process stops as soon as an equilibrium is reached, no more acceptable transactions can be made among the agents, or a time limit is reached. Each agent has a list of neighbors with whom he can communicate, e.g., an individual does not know everyone but the people of his social circle.

The definition of the neighborhood depends on the application context: a common representation is done by means of a graph, often called contact network, where nodes correspond to agents and a link between two nodes means that both agents know each other, e.g., (de Weerdt et al. [2012]). Hence, the neighborhood of an agent becomes the set of one-hop nodes. Contrary to what happens in a centralized system, where the auctioneer has complete information to determine the best transaction(s) to perform, in a distributed system, an agent keeps only up-to-date information about his neighborhood. Thus, the number of agents that can be involved in a transaction is limited to the neighborhoods of the involved agents.

Most studies consider only bilateral transactions, e.g., (Park and Yang [2007]). Even in a distributed framework, multilateral transactions remain difficult to compute with scalable means. As a result, approximations have been suggested, e.g., (Andersson and Sandholm [1998]). More work is needed to identify the utility functions and, more generally, the multi-agent systems, for which an optimal, or a near-optimal solution can be reached using a given set of transactions.

3.2.2 Automated Mechanism Design

Mechanism design is a sub-field of microeconomics and game theory that considers how to design the rules of interactions in a game to achieve specific properties. It suits especially problems that involve a set of self-interested agents, each with private information about their preferences. For example, a selfish agent can misreport information about his preferences to manipulate the mechanism and increase his profit. A proper design of the interaction rules can incite the agents to report only truthful information.

Mechanism design became a tool in computer science and operational research (Conitzer and Sandholm [2002]), due to distributed systems, like the Internet, which have many characteristics of an economy. Some studies (Sandholm [2003]) introduced the approach of Automated Mechanism Design (AMD). This latter approach tackles the mechanism design as an optimization problem, and is such that the mechanism is automatically created, using optimization algorithms, for the multi-agent system, with the definition of a specific objective or of a particular equilibrium.

To achieve a given objective or equilibrium, all properties must be explicitly translated into constraints in an optimization model.

If agents are self-interested, they reject all transactions that are not profitable for them as they aim at obtaining a greater utility than the one they had initially. However, if the mechanism satisfies the individual rationality constraints (by taking part in the mechanism, a player cannot receive less than what he could have obtained on his own), all involved agents in a transaction will accept to take part in it. To avoid manipulations of the mechanism by selfish agents, the mechanism must incite the report of the truth. Once the mechanism has been offered, the agents look at which preferences would give them the greatest utility. If an agent finds out that, by lying, he obtains a greater utility, the agent may lie as a consequence of his selfishness. However, if the mechanism satisfies the incentive compatibility constraints (every participant can achieve the best outcome to themselves just by acting according to their true preferences), the agent then reports truthfully his preferences.

Some authors, e.g., (Conitzer and Sandholm [2004]) managed to design AMD algorithms in the context of the bartering problem with bilateral exchanges and no compensatory payment. The algorithms work well for instances involving two agents with up to 90 types but only around 30 outcomes. Indeed, the complexity of determining the optimal mechanism grows quickly with the number of involved agents per transaction. More recent results attempt to use machine learning tools, e.g., (Narasimhan et al. [2016]).

3.2.3 Our Contributions

We first revisit the linear programming mathematical model that has been proposed for general multilateral transactions (either one-to-many or many-to-many) (Parkes [2001]) and re-interpret it as a decomposition model, i.e., a column generation model. It allows for a much more efficient solution scheme as it avoids the explicit enumeration of the outcomes.

In Operations Research, efficient tools have been developed over the last forty years to handle large scale optimization problems such as the column generation technique (Chvatal [1983]). Indeed, column generation has proved extremely efficient to solve linear programs with an exponential number of variables. It does not require to consider explicitly all columns (i.e., outcomes) to guarantee that the optimum has been reached.

We next propose a new linear programming model with a polynomial number of variables, and show that the first model corresponds to a Dantzig-Wolfe decomposition of the second, and provide a new proof of the polynomial complexity of AMD for many-to-many transactions.

Extensive computational results are provided for assessing the scalability of both linear programming models.

The paper is organized as follows. We state formally the automated mechanism design problem in Section 3.3. We then recall in Section 3.4.1 the first linear programming model for multi-agent transactions. In Section 3.4.2, we describe the alternate new linear programming formulation. Equivalence of the two models is demonstrated in Section 3.4.3. In Section 3.5, we discuss solution processes and their complexities. Extensive computational results are discussed in Section 3.6. Conclusions are drawn in the last section.

3.3 Problem Statement

Following Sandholm (Sandholm [2003]), the probabilistic automated mechanism design problem can be stated as follows. We assume that we are given: *(i)* a set of outcomes $\mathcal{O}$, indexed by O; *(ii)* a finite set of n agents A, indexed by a ; *(iii)* for each agent $a \in A$: *(iii-a)* a finite set Θ_a of t_a types, where each type $\theta_a^i \in \Theta_a$ defines a preference relationship among the outcomes, *(iii-b)* a probability distribution p_a over Θ_a, where p_a^θ is the probability that agent a reports type θ, for $\theta \in \Theta_a$, *(iii-c)* an additive utility function $u_a : \Theta_a \times \mathcal{O} \rightarrow \mathbb{R}$; *(iv)* a set Θ indicating the set of all types reported by all agents ; *(v)* a set Θ_{-a} which indicates the preferences of all other agents except agent a in the reported type profile Θ ; *(vi)* an objective function. In this study, we consider the utilitarian social welfare function (SW) as mathematically defined in (3.1) ; *(vii)* the probability g_θ^O of choosing outcome O when type profile $\theta \in \Theta$ is reported ; *(viii)* A set $\hat{\Theta} \in \Theta$, called the truthful type profile, in which each agent a has reported their truthful type $\hat{\theta}_a$.

Let R (indexed by r) be a finite set of m available resources in the multi-agent system. These resources are initially distributed over the multi-agent system. Let R_a^{INIT} be the initial bundle of m_a resources of agent a. The initial outcome is then defined as follows: $O^{\text{INIT}} = \{R_a^{\text{INIT}} : a \in A\}$.

The utility function is defined as usual, based on outcomes. Besides the selfish behavior of the agents, a social objective function is defined. Choices resulting from the social welfare theory are the most widely used for a global evaluation of a multi-agent system, e.g., utilitarian social welfare, elitist social welfare, egalitarian social welfare or Nash product. In the present study, we consider the utilitarian social welfare objective, denoted by SW, which sums the utility of each individual in order to obtain an overall welfare. All agents are treated the same, regardless of their initial level of utility. We then have:

$$\text{SW} = \sum_{a \in A} \sum_{\theta \in \Theta_a} \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O\,\underbrace{(\theta_a, \theta_{-a})}_{\theta})g_\theta^O, \tag{3.1}$$

where, by writing $O(\theta_a, \theta_{-a})$, we want to stress that O is the outcome when agent a reports type θ_a and other agents report types θ_{-a} in the reported type profile $\theta \in \Theta$.

3.4 Mathematical Models for the Probabilistic AMD Problem

We first recall the linear programming model, call CG_AMD, as proposed by Sandholm (Sandholm [2003]) with an exponential number of variables, and then a new linear programming model, called COMPACT_AMD with a polynomial number of variables. We next show that the two models are equivalent.

3.4.1 CG_AMD Model

The CG_AMD Model is defined over a probability distribution for each type profile θ, where the variables g_θ^O define the probability for the mechanism to choose outcome O when type profile $\theta \in \Theta$ is reported.

The set of constraints is written as follows, assuming a social welfare objective, as expressed in (3.1).

$$\sum_{O \in \mathcal{O}} g_\theta^O = 1 \qquad \theta \in \Theta \tag{3.2}$$

$$\sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a}))\, g_\theta^O \geq u_a(\hat{\theta}_a, O_{\text{INIT}}) \qquad \theta \in \Theta, a \in A \tag{3.3}$$

$$\sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\hat{\theta}_a, \hat{\theta}_{-a}))\, g_{\hat{\theta}}^O \geq \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \hat{\theta}_{-a})))\, g_\theta^O \qquad \hat{\theta}, \theta \in \Theta, a \in A, \tag{3.4}$$

$$g_\theta^O \geq 0 \qquad \theta \in \Theta, O \in \mathcal{O}. \tag{3.5}$$

Constraints (3.2) express the probability distribution for each type profile. In order to ensure the agents to be rational, a set of *individual rationality* (IR) constraints are added to the model. There are three different possible IR constraints: ex ante, ex interim, and ex post, depending on what the agent knows about its own type and the others' types when deciding whether to participate in the mechanism. We consider here the so-called interim ones, and they are translated into Constraints (3.3). Interim constraints incite the agents to take part in the transaction, and deal with the expected utility, i.e., the average utility that is obtained over the type profiles. Constraints (3.4) ensure that the mechanism is incentive compatible by applying the set of constraints satisfying the Bayes-Nash equilibrium. Let $\hat{\theta}$ be the truthful type profile, in which each agent reports truthful information, and $\hat{\theta}_a$ the truthful type of agent a. For the Bayes-Nash equilibrium, reporting truthful information gives on average the agents an equal or a greater utility, assuming the other agents also report truthful information which is being shown by $\hat{\theta}_{-a}$. Θ_{-a} denotes an incomplete type profile (see item *(v)* in the statement of the problem), without the reported type by agent a. Hence, $u_a(\hat{\theta}, O(\theta_a, \theta_{-a}))$ corresponds to the evaluation by agent a of outcome O, when agent a reports truthful information $\hat{\theta}$, and the other agents report any information θ_{-a} (truthful or untruthful types).

3.4.2 Model COMPACT_AMD

We now propose a new linear programming model, called COMPACT_AMD, with a polynomial number of variables. It uses one set of variables p such that p_{ar}^{θ} is equal to the probability of assigning resource r to agent a when type profile θ is reported. Model COMPACT_AMD is written as follows.

$$\max \sum_{a \in A} \sum_{\theta \in \Theta} \sum_{r \in R} u_{ar}^{\theta} \, p_{ar}^{\theta} \tag{3.6}$$

subject to:

$$\sum_{r \in R} u_{ar}^{\theta} p_{ar}^{\theta} \geq \sum_{r \in R_{\text{init}}^{a}} u_{ar}^{\theta} \qquad \theta \in \Theta, a \in A \tag{3.7}$$

$$\sum_{r \in R} u_{ar}^{\hat{\theta}} p_{ar}^{\hat{\theta}} \geq \sum_{r \in R} u_{ar}^{\theta} p_{ar}^{\theta} \qquad \theta, \hat{\theta} \in \Theta; a \in A \tag{3.8}$$

$$\sum_{a \in A} p_{ar}^{\theta} = 1 \qquad r \in R, \theta \in \Theta \tag{3.9}$$

$$p_{ar}^{\theta} \geq 0 \qquad \theta \in \Theta, a \in A, r \in R. \tag{3.10}$$

Constraints (3.7) express the individual rationality constraints. Constraints (3.8) ensure that the mechanism is incentive compatible using the Bayes-Nash equilibrium constraints. Constraints (3.9) guarantee the probability distributions.

3.4.3 Equivalence of Models CG_AMD and COMPACT_AMD

Theorem 3.1. *Models* CG_AMD *and* COMPACT_AMD *are equivalent.*

Proof. Proof consists in re-interpreting Model CG_AMD as a so-called column generation model, and Model COMPACT_AMD as the compact model associated with CG_AMD, i.e., Model CG_AMD can be derived from COMPACT_AMD using a Dantzig-Wolfe decomposition. Reader who is not familiar with linear programming and decomposition techniques in mathematical programming is referred to (Lasdon [1970]).

Dantzig-Wolfe decomposition relies on the Minkowski theorem, which states that all bounded polyhedra P are convex combinations of their extreme points $\text{EXTR}(P)$, i.e., any point x of P can be written $x = \sum_{e \in \text{EXTR}(P)} \lambda_e e$ with $\sum_{e \in \text{EXTR}(P)} \lambda_e = 1$ and $\lambda_e \geq 0$.

As the polyhedron $P^{\text{COMPACT\_AMD}}$ defined by the set of Constraints (3.7)-(3.10) of Model COMPACT_AMD is bounded thanks to Constraints (3.9), we use Minkowski's theorem for polytopes (i.e., bounded polyhedrons). Let $E = \{e_j^{ar\theta} : j \in J\}$ be the set of extreme points of $P^{\text{COMPACT\_AMD}}$, where J is the index set of the extreme points. Then, for any $(a, r, \theta) \in A \times R \times \Theta$, there exists a set of scalars (λ_j^θ) such that we can rewrite each p_{ar}^θ in the following way:

$$p_{ar}^\theta = \sum_{j \in J} \lambda_j^\theta e_j^{ar\theta} \qquad a \in A, r \in R, \theta \in \Theta, \tag{3.11}$$

with $\sum_{a \in A} e_j^{ar\theta} = 1$ for $r \in R, j \in J, \theta \in \Theta$. After substituting p_{ar}^θ by the convex combination of its extreme points using equation (3.11) in Constraints (3.7)-(3.10), and after re-interpreting each outcome O as an extreme point e_j, we can then establish a one-to-one correspondence between (3.7)-(3.10) and (3.2)-(3.4).

For example, consider Constraints (3.9). Using the decomposition into extreme points for given r and θ, constraints (3.9) can be rewritten:

$$\sum_{a \in A} p_{ar}^\theta = \sum_{a \in A} \left(\sum_{j \in J} \lambda_j^\theta e_j^{ar\theta} \right) = \sum_{j \in J} \underbrace{\left(\sum_{a \in A} e_j^{ar\theta} \right)}_{\text{equal to 1}} \lambda_j^\theta = \underbrace{\sum_{j \in J} \lambda_j^\theta}_{\sum_{O \in \mathbb{O}} g_\theta^O} = 1 \tag{3.12}$$

which is equivalent to (3.2). Similar calculations allow the derivation of the other constraints of Model COMPACT_AMD from the constraints of Model CG_AMD. □

3.5 Solution Process

After the re-interpretation of CG_AMD as a decomposition model, we now explain how to solve it efficiently, using a so-called column generation algorithm.

3.5.1 Column Generation Framework

The CG_AMD model corresponds to a Linear Program (LP) with a very large number of variables, which is difficult to solve, using the classical Simplex algorithm. We, therefore, propose to consider the Column Generation (CG) technique to solve it, i.e., a linear programming tool dedicated to large scale linear programs, e.g., (Chvatal [1983]) if not familiar with column generation techniques.

Column Generation relies on the fact that only a very small fraction of the variables are nonzero in the optimal linear programming solution. In fact, it will never be more than the number of constraints, according to linear programming theory (Chvatal [1983]). Hence, only a subset of columns must to be explicitly enumerated when solving a CG model, typically the columns/-variables that takes a nonzero value in the optimal LP solution, with the challenge of identifying them. This property is true for the CG_AMD model as it involves an exponential number of variables due to the combination of different types and outcomes, but a polynomial number of constraints.

CG consists in solving alternatively a restricted master problem (i.e., the CG_AMD model in Section 3.4.1 with a very limited number of columns/variables) and the pricing problem (generation of a new outcome associated with a given set of types, therefore the pricing problem is indexed by θ and O) until the optimality condition is satisfied (i.e., no outcome/type configuration with a negative reduced cost). In other words, when a new outcome/type configuration is generated, it is added to the current restricted master problem only if its addition results in an improvement of its current optimal value. This condition, indeed an optimality condition, can be checked with the sign of the reduced cost, denoted by $\overline{c}_{\theta O}$, see below for its expression in (3.13) (the reader who is not familiar with linear programming concepts is referred to (Chvatal [1983])).

The column generation algorithm is depicted in Figure 3.1. Each pricing problem (generation of a new outcome/type configuration) is denoted by PP$_{\theta O}$. Its primary variables are defined as follows: $x_{ra} = 1$ if agent a owns resource r, and 0 otherwise.

Decomposing the utility of each agent over the set of resources using the expression of the reduced cost is written as follows:

$$\overline{c}_{\theta O} = c_{\theta O} - w^{\text{PROBA}}_{\theta} - \sum_{a \in A} \Bigg[\Big(w^{\text{IR}}_{a,\theta} + \sum_{\substack{\theta' \in \Theta \\ \theta' \neq \theta}} w^{\text{IC}}_{a,\theta,\theta'} \Big) \sum_{r \in R} u_a(\theta, r)\, x_{ra} - \sum_{\substack{\theta' \in \Theta \\ \theta' \neq \theta}} w^{\text{IC}}_{\theta,\theta',\theta} \sum_{r \in R} u_a(\theta', r)\, x_{ra} \Bigg] \tag{3.13}$$

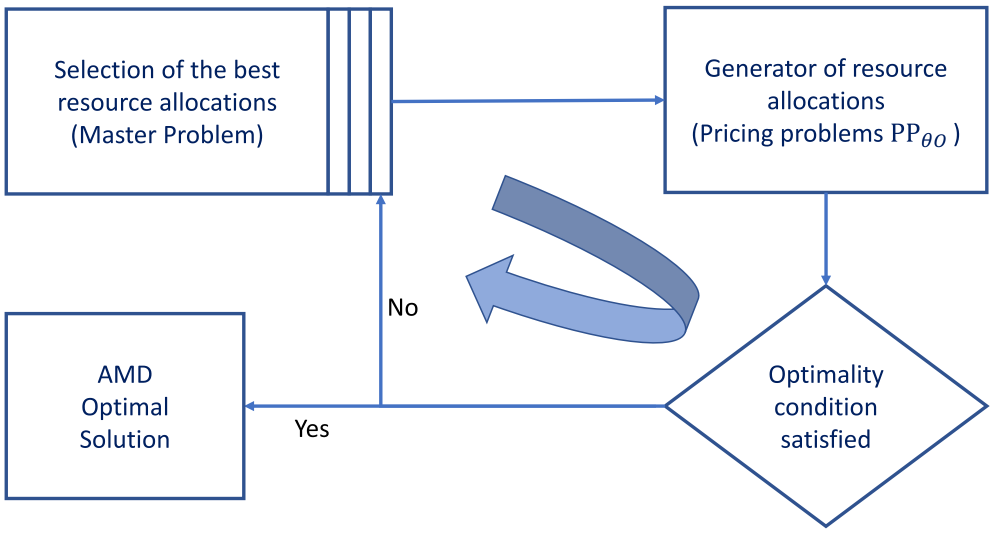

Selection of the best resource allocations (Master Problem)
Generator of resource allocations (Pricing problems PP$_{\theta O}$)
No
Optimality condition satisfied
AMD Optimal Solution
Yes

FIGURE 3.1: Column generation algorithm

where $w_{\theta}^{\text{PROBA}}$, $w_{a,\theta}^{\text{IR}}$, $w_{a,\theta,\theta'}^{\text{IC}}$ are the values of the dual variables associated with Constraints (3.2), (3.3), and (3.4), respectively, and where

$$c_{\theta O} = \sum_{a \in A} \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a})). \tag{3.14}$$

In the case of many-to-many transactions among the agents, the constraint set of the pricing problem is limited to expressing that a resource can be owned by one agent at a time:

$$\sum_{a \in \mathcal{A}} x_{ra} = 1 \qquad r \in R. \tag{3.15}$$

3.5.2 Polynomial Complexity

We next analyze the complexity of the column generation solution process.

Theorem 3.2. *For many-to-many transactions, the pricing problem of the Model* COMPACT_AMD *can be solved in polynomial-time for the utilitarian social welfare objective.*

Proof. We will show that the pricing problem can be solved in polynomial time. Let us define w_{ar} as follows:

$$w_{ar} \quad = \quad (w_{a,\theta}^{\text{IR}} + \sum_{\substack{\theta' \in \Theta \\ \theta' \neq \theta}} w_{a,\theta,\theta'}^{\text{IC}}) \ \ u_a(\theta, r) \ - \sum_{\substack{\theta' \in \Theta \\ \theta' \neq \theta}} w_{\theta,\theta',\theta}^{\text{IC}} u_a(\theta', r). \tag{3.16}$$

The objective function of the pricing problem ($\text{PP}_{\theta O}$) can then be rewritten as follows:

$$\max \sum_{a \in A} \sum_{r \in R} w_{ar} x_{ar} - w_{\theta}^{\text{PROBA}}, \tag{3.17}$$

where

$$w_{ar} = c_{\theta O} - w_a \times \alpha_{\theta O}^{\text{IR}} - w_{a,\theta',\theta}^{\text{IC}} \times \alpha^{\text{IC}}. \tag{3.18}$$

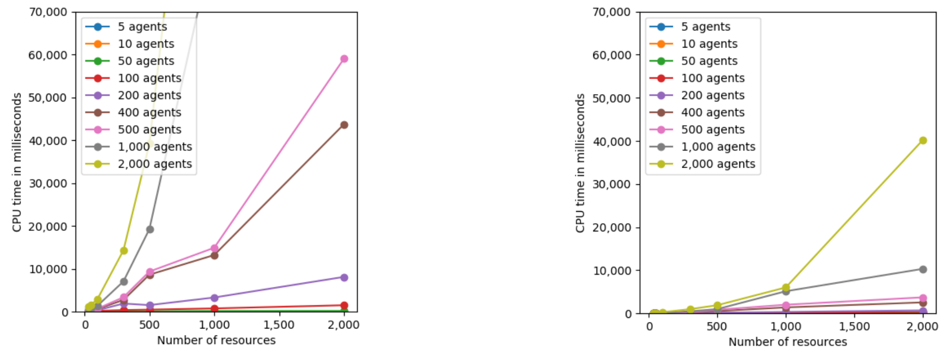
Observe that each pricing problem $\text{PP}_{\theta O}$ can be decomposed into $|A|$ independent sub-problems $\text{PP}\,_{\theta O}^{r}$ such that :

$$\max \left\{ \sum_{a \in A} w_{ar} x_{ar} : \sum_{a \in A} x_{ar} = 1, x_{ar} \in \{0,1\} \text{ for } a \in A \right\}. \tag{3.19}$$

Optimal value of $\text{PP}\,_{\theta O}^{r}$ is therefore $\max_{a \in A} w_{ar}$ with $x_{\tilde{a}r} = 1$ for $\tilde{a} = \arg\max_{a \in A} w_{ar}$ and all the others $x_{ar} = 0$.

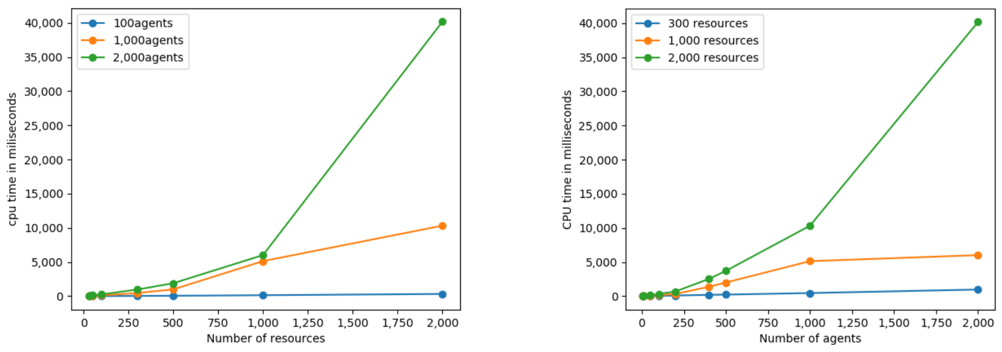
It follows that the pricing problem (outcome/type generator) can be solved in $O(n.|R|)$, where $n = |A|$. $\square$

Considering Model COMPACT_AMD, it has a polynomial number of variables and constraints. As linear programming is in P, it means that the compact formulation associated with the column generation Model CG_AMD can be solved in polynomial time, or very efficiently using any Simplex algorithm.



(a) CG_AMD Model (b) COMPACT_AMD Model

FIGURE 3.2: Computing Time Comparison



(a) Number of Resources (b) Number of Agents

FIGURE 3.3: Sensitivity Analysis

3.6 Experimental Results

We implemented both optimization models CG_AMD and COMPACT_AMD, solving their linear (integer) programming (sub) problems using the CPLEX solver (Cplex [2014]). We first describe the generation of the data sets, and next compare the computational performance of the two models. We go on with a sensitivity analysis with respect to the number of agents and resources for the best of the two models.

3.6.1 Data sets

We generated random datasets, parameterized by the number of agents and resources. As with other results reported in the literature, we also only consider one type per agent. For each agent and resource assignment, the preference of the agent towards that resource was generated randomly as an integer number $\{1..10\}$. For each data set, the initial resource allocation was computed to maximize the social welfare of the system.

For each set of parameters, we generated 10 instances, and build the plots with averages over 10 data sets.

3.6.2 Comparison of the Two Models

To compare the two optimization models, we solve both of them on the same data sets with an increasing number of agents, ranging from 5 up to 2,000, while the number of resources ranges from 20 to 2,000. Computational times are reported in milliseconds in Figure 3.2.

For the CG_AMD, we solve data instances up to 2,000 agents and 2,000 resources in around 800 seconds, while for the COMPACT_AMD, we solve them in in less than 42 seconds. In Figure 3.2, the horizontal axis denotes the number of resources and the vertical axis indicates the average of the CPU times taken over ten different data sets with each color corresponding to a different number of agents.

Out of the 10 data sets for the largest data set, i.e., 2,000 agents and 2,000 resources, the smallest computational time for solving model CG_AMD was 744 seconds and the largest was 784 seconds while for the COMPACT_AMD, for the same data set, the smallest computational time was 33 seconds and the largest computational time was 46 seconds.

Taking a close look at Figure 3.2(a), one can observe that in some cases the CG_AMD model takes more time to solve some instances of a smaller size than an instance of a larger size (in terms of the number of agents and resources). Because of the sensitivity to the initial resource allocation for the different instances, i.e., to the set of initial columns used to solve the CG_AMD model.

Model COMPACT_AMD surpasses model CG_AMD in terms of computational times whether we consider an increasing number of resources or an increasing number of agents, as illustrated in Figure 3.2. Although Model CG_AMD is slower than COMPACT_AMD in the case of many-to-many transactions, its interest lies in its versatility in solving AMD for different types of transactions such as one-to-many and many-to-one transactions. Indeed, the pricing problem can accommodate more complex constraints as the ones defining those latter transactions, this will be studied in future work.

3.6.3 Sensitivity Analysis

Growth in either the number of agents or resources has a very strong effect on the average computational times of the COMPACT_AMD model. More specifically, as one can observe in Figure 3.3(b), there is a significant increase in the average CPU time as the number of agents increases regardless of the number of resources in the system.

An increase in the number of resources has even a stronger impact on the average CPUs of the COMPACT_AMD model especially when the number of resources is much closer to the agents, see Figure 3.3(a).

3.7 Conclusion

We designed a first highly time scalable algorithm for Automated Mechanism Design (AMD) in the probabilistic setting with no payment. It uses a decomposition framework with a pricing problem that can be solved in polynomial time for many to many transactions.

Moreover, we exhibit a second formulation (COMPACT_AMD model) with a linear program and a polynomial number of variables and constraints, a straightforward way to demonstrate the polynomial-time complexity of the AMD problem, assuming the use of polynomial time algorithms for linear programming (Khachiyan [1980]).

The COMPACT_AMD model is more time efficient than the CG_AMD one, and data sets with up to 2,000 single-type reporting agents and 2,000 resources can be solved in about 34 seconds.

Future work will involve the solution of AMD models with different transaction types, for which the CG_AMD model may be easier to adapt. We also plan to investigate different types of trades such as the Vickey-Groves (VCG) one, or AMD models with payment features.

Chapter 4

A Multilateral Transaction Design with a Large Scale Optimization Tool

This chapter is based on the second paper that we have worked on during my master of computer science degree. It corresponds to a journal paper, extending some of the models already developed in the conference paper of Chapter 3.

In the paper defining this chapter, we exploited the structure of the column generation model (CG_AMD) that was introduced in Chapter 3 to design three variants of model CG_AMD, each associated with a different type of trade among the agents, i.e., many-to-one trades, one-to-one trades and many-to-many trades. We then used the column generation algorithm to solve the resulting models and compare them in terms of objective value and computational times.

We showed that by solving the two equivalent models corresponding to the many-to-many trades (CG_AMD and COMPACT_AMD), one could not only reach better solutions than the ones resulting from the solutions of the models associated with many-to-one and one-to-one trades, but also could solve the many-to-many models much faster either using the column generation algorithm for the CG_AMD model or using the simplexe algorithm to solve the COMPACT_AMD model.

4.1 Abstract

Mechanism design deals with making social decisions in multi-agent systems. Using linear programming tools to find the best mechanisms leads to Automated Mechanism Design (AMD). It can be used to solve the problem of multi-agent resource allocation when the agents are self-interested and rational, and each agent has some private information/preference for the resources in the system. AMD consists of designing the rules of interaction among these agents to reach an equilibrium despite the selfish behavior of these agents.

The current literature model for the AMD problem faces immediate scalability issues as it contains an exponential number of variables. By re-interpreting the existing literature model as a decomposed model, we proposed an efficient column generation algorithm to solve the AMD problem subject to different types of transactions: one-to-one, one-to-many, and many-to-many, leading to three variants of the AMD column generation model. We also investigated the expression of the compact model for the many-to-many trades, i.e., the model such that applying the Dantzig-Wolfe decomposition on it would lead to the so-called column generation model in the case of many-to-many transactions. The compact model involves a polynomial number of variables, and provides us with a new proof for the polynomial-time solution of AMD subject to many-to-may transactions.

New models and algorithms are then thoroughly tested. Numerical results show that significantly larger data instances than those reported in the literature can be solved. Indeed, we can solve the AMD problem for up 2,000 agents or 2,000 resources for the many-to-many/multilateral trades, 1,000 agents and 1,000 resources for the many-to-one trades and 50 agents and 300 resources for the bilateral trades.

4.2 Introduction

The multi-agent resource allocation problem has been studied in different fields of science such as Economics, Computer Science or Operations Research. The problem of allocating a set of indivisible goods among a set of self-interested rational agents can be modeled using linear programming and will immediately face scalability issues as the number of agents or the number of resources increases. This mainly happens due to the exponential number of variables of the problem.

One can tackle the multi-agent resources problem via a centralized approach, in which the agents will report their preferences to a central entity called the auctioner (which could be one of the agents as well) and this entity will determine the winner of the auction. An example is, e.g., the winner determination problem in combinatorial auctions (Sandholm [2002]). One can solve the problem using a distributed approach by letting agents distribute their initial set of resources among themselves by allowing local negotiations until an equilibrium is reached. The advantage of the second approach is that when the centralized entity has very limited computational power or is not trustworthy, the system can still evolve towards reaching an equilibrium using negotiations among the agents (Chevaleyre et al. [2017]).

Finding an optimal or near optimal solution after a sequence of negotiations is still a critical issue as it is difficult to find the most promising transactions among more than two agents. Efficient resource allocation is a complex issue and faces scalability issues very quickly.

Different types of transactions can be allowed depending on the application context or the topology of the social network. In some networks an agent can connect to all other agents hence the network is strongly connected whereas in some other networks each agent can only trade its resources with a limited number of agents and not all of them.

The effect of different types of transactions has been analyzed in (T.Sandholm [1998]) in a theoretical point of view. The paper starts with the simplest transactions, i.e., the original, cluster and swap contracts, which are basically one-to-one (bilateral) transactions and allow each agent to trade with exactly one other agent at a time, and continues with the multi-agent (multilateral) contracts, which allow groups of agents to negotiate their set of resources among themselves. When describing the multilateral contracts, the author introduces the so called OCSM contracts which are a combination of all of the contract (trade) types introduced in that paper and allow any of the agents to trade any number of resources in their possession with other agents in their neighbourhood, i.e., many-to-many transactions. Sandholm (T.Sandholm [1998]) also showed that multilateral transactions are sufficient to guarantee reaching the optimal resource allocation. Experimental results are discussed in (M.R.Andersson and T.Sandholm [1998]) with a comparison of the different types of contracts introduced in (T.Sandholm [1998]) and their effect on the objective function and their computational times.

Very few studies have been carried out on many-to-many transactions due to the difficulties of designing and implementing them using scalable algorithms (Friedman and Rust [1993]). Different negotiation frameworks and the effect of the type of deals have been discussed in (Endriss et al. [2006]). Indeed, different types of deals and restrictions will affect the scalability of the problem hence some studies suggest to restrict the transactions to some simple ones and only use more complex transactions when the simpler ones are not being helpful (Andersson and Sandholm [1998]). An approach that usually leads to a local optimum.

Another parameter that affects the scalability as well as the social welfare of the system is the modularity of the valuation/utility functions. This property can simply be viewed as how much the agent values a bundle of resources with respect to the values of each of the elements in the bundle. If the agent values a bundle equally as the sum of the values of each of the resources in the bundle, the valuation function is called modular. If the agent values the bundle less than or equal to the sum of each of the resources in the bundle, the valuation function is called sub-modular and if the agent values the bundle higher than the sum of all of the resources in the bundle, the valuation function is called super modular. Different types of valuation functions have been studied in the literature. In (Chevaleyre et al. [2017]) the authors have discussed the effect of modularity of the valuation functions on the outcome of the mechanism with respect to envy free-ness and fairness.

Some authors showed that by using qualitative measures for the agent preferences towards the resources, it is possible to reach a socially optimum allocation (Chevaleyre et al. [2008]). Different resource allocations and social optima might be reached by allowing different types of trades and by setting restrictions on the types of transactions, see (Chevaleyre et al. [2007]) and (Estivie et al. [2006]). However, most of these studies have focused on bilateral trades and have not presented any proof regarding the convergence to the optimal resource allocation. A recent survey by (Ye et al. [2017]) discusses both the theoretical challenges and the various applications of self-organized multi-agent systems. Some other studies (Chevaleyre et al. [2005]) focused on particular cases where the authors either obtain a convergence property or a decrease of the theoretical complexity.

4.2.1 Distributed Multi-agent Systems

The multi-agent resource allocation problem is defined over a set of autonomous agents, that locally negotiate their resources with other agents in their neighbourhood. The negotiation process continues until, either an equilibrium is reached or a certain time limit has elapsed. Each agent can communicate with a set of agents in its neighbourhood. The definition of the neighborhood depends on the application context. The network representation can be done in different ways for which the most common one is the graph representation. In this representation each agent in the network corresponds to a node in the graph and if there exists an edge between two nodes/agents it is an indicator that the agents know each other, e.g., (de Weerdt et al. [2012]). Hence, the neighborhood of an agent becomes the set of one hop nodes. Contrary to what happens in a centralized approach, where a central entity needs to have complete information about the system in order to preform the best set of transactions, in a distributed system, agents only need to keep up-to-date information of their neighborhood. Thus, the number of agents that can be involved in a transaction is limited to the neighborhoods of the involved agents. Most of the focus of literature has been on bilateral transactions. Even in a distributed framework, due to their scalability issues, multilateral transactions remain difficult to compute. As a result, some studies have illustrated approximation alogrithms to solve the problem see, e.g., (Andersson and Sandholm [1998]). Most of the so-called multilateral negotiation systems, as in (Park and Yang [2007]), are based on using a sequence of bilateral transactions. It is worth noting that, even if transactions are restricted to bilateral transactions, it does not always prevent the mechanism from reaching an optimal resource allocation, see, e.g., (Nongaillard et al. [2008]). More work is needed in order to identify the multi-agent systems for which an optimal, or a near optimal solution can be reached using a restricted set of transactions such as, e.g., bilateral transactions or one-to-many transactions.

4.2.2 Automated Mechanism Design

Mechanism design, a sub-field of microeconomics and game theory, is concerned with designing the rules of interaction among a set of self interested rational agents in a game to achieve some specific properties. It suits especially cases for which each agent has some private information about their preferences and has incentive to lie or hide them. For instance, a selfish agent might misreport their preferences in order to manipulate the outcome of the mechanism to their advantage. In such cases a proper design of the interaction rules can incite the agents to report only truthful information.

Due to the growth of distributed systems like internet and their applications in economy, Mechanism Design became a tool in Computer Science and Operational Research (Conitzer and Sandholm [2002]). In (Sandholm [2003]) the author introduced an approach to solve the mechanism design problem called the "Automated Mechanism Design (AMD)" in which the mechanism design problem is viewed as an optimization problem. Hence the mechanism will be created and solved using optimization algorithms to reach the objective at hand.

In order to reach an equilibrium, all properties of the mechanism design problem such as the self-interested behavior or the rationality of the agents have to be explicitly translated into constraints of the optimization model. For example, a self interested, rational agent will reject all transactions that are not beneficial for it in order to gain a greater utility than what it had initially. This can be done if the mechanism satisfies a set of constraints that will ensure if an agent takes part in the mechanism it will be better off than if it were on its own. A property referred to in the literature as the Individual Rationality of the mechanism. A mechanism might also hinder a set of agents from misreporting their information by applying a set of constraints called the incentive compatibility constraints. Indeed an agent might misreport its information if it were the case that by lying it could increase its utility.

In the past some authors (see, e.g., (Conitzer and Sandholm [2004])) introduced some AMD algorithms for specific cases such as the bartering problem by only allowing bilateral exchanges with no compensatory payment. The algorithms solves the problem for instances involving two agents with up to 90 types but only around 30 outcomes. Indeed, there is a direct relation between the complexity of determining the optimal answer and the number of involved agents in a transaction. More recent results attempt to use machine learning tools, see, e.g., (Narasimhan et al. [2016]).

4.2.3 Our Contributions

We first revisit the mathematical model proposed in (Parkes [2001]) and re-interpret it as a decomposed model which can be solved using the Column Generation algorithm e.g. see (Chvatal

[1983]). A technique that has proven to be extremely efficient to solve linear programs with an exponential number of variables. The Column Generation algorithm will help us with an efficient solution scheme by allowing to skip the explicit enumeration of the entire set of outcomes(columns) and only enumerate those which would result in an improvement in the objective value.

We will then, take advantage of the versatility of the structure of the sub problem associated with the decomposed model, i.e., the so-called pricing problem in Operations Research, and design three variations of the Automated Mechanism Design problem with respect to different types of trades between the agents such as many-to-many, many-to-one and one-to-one types of trades. We then, will show that the pricing problem associated with the many-to-many model which is actually more encountered in practice can be solved in polynomial time.

After, we will illustrate a new model to solve the many-to-many types of trade and will show that the column generation model corresponding to the many-to many types of trade is the Dantzig-Wolf decomposition of the new model.

Computational results and a comparison between different models and algorithms in terms of the objective value and the computational time are provided at the end of the paper in order to show the scalability of our algorithms. Indeed, using our models we were able to solve the AMD problem for data sets significantly larger than the ones existing in the literature,i.e. up to 2,000 agents and 2,000 resources.

The paper is structured as follows: A formal statement of the problem is presented in section 4.3. The column generation algorithm and our models for different types of trades are presented in section 4.4. The experimental results and a comparison between the models are presented in section 4.5 and the conclusion and future work are discussed in section 4.6.

4.3 Problem Statement

Following Sandholm‘s (Sandholm [2003]) work, the automated mechanism design reformulation as an optimization problem is defined as follows.

We assume that we are given:

(i) a set of outcomes $\mathcal{O}$, indexed by O,

(ii) a finite set of n agents A, indexed by a,

(iii) for each agent $a \in A$,

- a finite set Θ_a of t_a types, where each type $\theta_a^i \in \Theta_a$ defines a preference relationship among the outcomes,
- a probability distribution p_a over Θ_a, where p_a^θ is the probability that agent a reports type θ, for $\theta \in \Theta_a$,
- an additive utility function $u_a : \Theta_a \times \mathcal{O} \to \mathbb{R}$.

(iv) a set Θ indicating the set of all types reported by all agents,

(v) a set Θ_{-a} which indicates the preferences of all other agents except agent a in the reported type profile Θ,

(vi) an objective function. In this study we consider the utilitarian social welfare function (SW) as mathematically defined in 4.1 whose expectation the designer wishes to maximize.

(vii) the probability g_θ^O of choosing outcome O when type profile $\theta \in \Theta$ is reported.

(viii) A set $\hat{\Theta} \in \Theta$ named the truthful type profile for which each agent a has reported their truthful type $\hat{\theta}_a$.

In the particular context of the resource allocation problem, let R be a finite set of m available resources in the multi-agent system. Assume R is indexed by r. These resources are initially distributed over the multi-agent system. Let R_a be the initial bundle of m_a resources of agent a. An outcome is then defined by a resource allocation, i.e., $O = \{R_a : a \in A\}$.

In this work we assume that agents have additive valuation towards the resources. Meaning that the the utility of agent a for a bundle of m resources R is the summation of its utility over each of the m elements in the bundle.

As usual, the utility function is defined based on the outcome. Besides the selfish behavior of the agents, a social objective function is defined and corresponds to the objective function of the optimization model expressing the automated mechanism design. There are multiple choices that can be considered and borrowed from the social welfare theory (Arrow [1963], Moulin [1988]).

These notions are most widely used for a global evaluation of a multi-agent system, e.g., the utilitarian social welfare, the elitist social welfare, the egalitarian social welfare or the Nash product. Various concepts are also derived from the Nash equilibrium for example if the mechanism has some agent initiator, it is possible to focus on the utility of the agent initiator.

In this study, we will only consider the utilitarian social welfare objective, denoted by SW, which sums the utility of each individual in order to obtain the society's overall welfare. All agents are treated equally, regardless of their initial level of utility. We then have:

$$\text{SW} = \sum_{a \in A} \sum_{\theta \in \Theta_a} \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O\,\underbrace{(\theta_a, \theta_{-a})}_{\theta})g_\theta^O, \tag{4.1}$$

where, by writing $O(\theta_a, \theta_{-a})$, we want to stress that O is the outcome when agent a having truthful type $\hat{\theta}_a$ reports type θ_a and other agents report types θ_{-a} in type profile $\theta \in \Theta$.

Side-payments can be permitted or not. Usually payments characterize the set of possible trades that two rational agents would be willing to get involved in. Indeed, by allowing payments in the mechanism, the evaluation of a resource allocation can be broken down in two parts: A valuation function for which agents evaluate a resource bundle according to their preferences, and a payment function which may compensate a utility loss from an agent that is involved in the transaction (Parkes [2001]). Hence with the presence of money in the mechanism both the valuation function and the payment function must be considered in order to determine whether or not two or more rational agents are willing to trade their resources. However, the use of payments only corresponds to an enlargement of the set of possible rational trades. In this work we will not discuss the use of payments hence the evaluation/utility function can be restricted only to the valuation function.

4.4 CG_AMD: A First Optimization Model

4.4.1 Deterministic vs. Probabilistic Models

Conitzer and Sandholm [2002] introduced two types of mechanism: deterministic and randomized. A deterministic mechanism is a function from the vector of reported types to the set of real numbers such that having the reported agent types, the function will output a single number. In their work, they have also shown that the deterministic mechanism design problem, is NP-complete regardless of whether it is implemented in the dominant strategies or in the Bayes-Nash equilibrium. A randomized mechanism defined on the vector of reported types produces a probability distribution over the outcome set and theoretically allow us to solve the problem in polynomial time.

In this study, we will focus on randomized mechanisms.

4.4.2 Probabilitic AMD as an Optimization Model

The optimization model is defined over a probability distribution for each type profile θ, where the variables g_θ^O define the probability of the mechanism choosing outcome O when type profile

$\theta \in \Theta$ has been reported. We therefore have:

$$\sum_{O \in \mathcal{O}} g_{\theta}^{O} = 1 \qquad \theta \in \Theta. \tag{4.2}$$

In order to ensure the agents to be rational, a set of *individual rationality* constraints are added to the model. We provide below the expressions of the so-called ex post and interim ones. Other variants can be found in (A. Mas-Colell and M.D. Whinston and J.R. Green [1995]).

Ex post: Each agent has to obtain an equal or a larger utility than what it had initially, once it knows about the outcome chosen by the mechanism. If there exists an outcome O for which an agent will have a utility less than its initial utility for a given type profile θ, the corresponding constraint forces the variable g_{θ}^{O} to be zero. Constraints corresponding to Ex-post deals are defined as follows:

$$u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a}))\, g_{\theta}^{O} \geq u_a(\hat{\theta}_a, O_{\text{INIT}}) \qquad a \in A,\ \theta \in \Theta,\ O \in \mathcal{O}. \tag{4.3}$$

Interim: Before learning about the outcome chosen by the mechanism, the expected utility of the agents is greater than or equal to, their initial utility. This interim scenario incites the agents to take part in the transaction. However, these constraints do not guarantee a greater utility. Indeed, constraints deal with the expected utility, i.e., the average utility that is obtained over the type profiles: This involves some uncertainty. Constraints are written as follows:

$$\sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a}))\, g_{\theta}^{O} \geq u_a(\hat{\theta}_a, O_{\text{INIT}}) \qquad a \in A,\ \theta \in \Theta\ O \in \mathcal{O}. \tag{4.4}$$

Related to the selfishness of the agents, a set of *incentive compatibility* constraints are added to the model. Bayes-Nash equilibrium or a dominant-strategy equilibrium constraints can be used. Let $\hat{\theta}$ be the truthful type profile, in which each agent reports truthful information, and $\hat{\theta}_a$ be the truthful type of agent a. In the case of a Bayes-Nash equilibrium, reporting truthful information gives the agents an equal or a greater utility, assuming the other agents also report truthful information which is being shown by $\hat{\theta}_{-a}$. Constraints are written as follows:

$$\sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\hat{\theta}_a, \hat{\theta}_{-a}))g_{\hat{\theta}}^{O} \geq \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \hat{\theta}_{-a}))\, g_{\theta}^{O} \qquad a \in A,\ \hat{\theta}, \theta \in \Theta. \tag{4.5}$$

In the case of a dominant-strategy equilibrium, reporting truthful information gives the agents at least an equal utility even if the other agents misreport their type. Constraints for a dominant-strategy equilibrium, which are stronger than the constraints for a Bayes-Nash equilibrium, can be written as follows:

$$\sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\underbrace{\hat{\theta}_a, \theta_{-a}}_{\theta'}))g^O_{\theta'} \geq \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a}))g^O_\theta$$

$$a \in A, \theta', \theta \in \Theta, \theta_{-a} \in \Theta_{-a}, \quad (4.6)$$

where Θ_{-a} is an incomplete type profile (see item (v) in the statement of the problem), without the reported type by agent a. Hence, $u_a(\hat{\theta}_a, O(\hat{\theta}_a, \theta_{-a}))$ corresponds to the evaluation by agent a of outcome O, when agent a reports truthful information $\hat{\theta}_a$, and the other agents report any information θ_{-a} (truthful or untruthful types). This means that regardless of what the other agents report, each agent would be better of staying truthful.

4.4.3 CG_AMD Model

There has been very few scalable algorithms and optimization tools developed to solve the Automated Mechanism Design problem. In most of the existing literature, the focus of the research is to find a scalable method to solve the problem with respect to the social welfare of the system. Following the existing work, in the sequel we are aiming to maximize the social welfare in the system using a tool borrowed from large scale optimization, i.e., the column generation algorithm.

The CG_AMD model, which is defined over the probability distribution for each type profile θ, can be written as follows:

$$\sum_{O \in \mathcal{O}} g^O_\theta = 1 \qquad \theta \in \Theta \qquad (4.7)$$

$$\sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a})))\, g^O_\theta \geq u_a(\hat{\theta}_a, O_{\text{INIT}}) \qquad \theta \in \Theta, a \in A \qquad (4.8)$$

$$\sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\hat{\theta}_a, \hat{\theta}_{-a}))\, g^O_{\hat{\theta}} \geq \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \hat{\theta}_{-a})))\, g^O_\theta \qquad \hat{\theta}, \theta \in \Theta, a \in A \qquad (4.9)$$

$$g^O_\theta \geq 0 \qquad \theta \in \Theta, O \in \mathcal{O}. \qquad (4.10)$$

Constraints (4.7) express the probability distribution for each type profile. Constraints (4.8) make sure that the agents are rational by applying the so called *Interim* constraints discussed in section 4.4.2. Constraints (4.9) correspond to the Bayes-Nash equilibrium constraints which are added to the model to ensure the mechanism to be incentive compatible.

4.4.4 Solution of the CG_AMD Model

The CG_AMD model corresponds to a linear program with a very large number of variables, which is therefore difficult to solve using the classical simplex algorithm. Hence we propose an algorithm borrowed from large scale optimization called the column generation algorithm which is used to solve problems with an exponential number of variables. see, e.g., (Chvatal [1983]) or (Lübbecke and Desrosiers [2005]) if not familiar with column generation techniques.

4.4.4.1 Column Generation Framework

Column Generation works based on the fact that in the optimal solution to the linear program, only a small number of the variables will take nonzero values which according to the theory of linear programming will never exceed the number of constraints (Chvatal [1983]). Hence the quest will be to identify those columns/variables that take nonzero values in the optimal LP solution. A property that also holds for the CG_AMD model as it consists of an exponential number of variables coming from the combination of different types and outcomes, yet having a polynomial number of constraints.

A column generation algorithm alternately solves a restricted master problem (the CG_AMD model in Section 4.4.3 starting with a limited number of columns/variables) and a so-called pricing problem, which generates a new potential outcome associated with a given type profile with an improved objective value. The process continues until the optimality condition is satisfied, i.e., no new column/outcome with a negative reduced cost can be found or in other words no new outcome with an improved objective value. When a new column is generated, it is added to the current set of columns in the restricted master problem only if by adding the column there will be an improvement in the objective value of the current restricted master problem. This condition can easily be validated using the sign of the reduced cost, denoted by $\bar{c}_{\theta O}$, which is defined in (4.12) (the reader who is not familiar with linear programming concepts is referred to (Chvatal [1983])).

The column generation algorithm is depicted in Figure 4.1. The expression of the reduced cost is described below it.

Each pricing problem is indexed by θ and O, and its primary variables are defined as follows:

$$x_{ra} = \begin{cases} 1 & \text{if agent } a \text{ owns resource } r \\ 0 & \text{otherwise.} \end{cases}$$

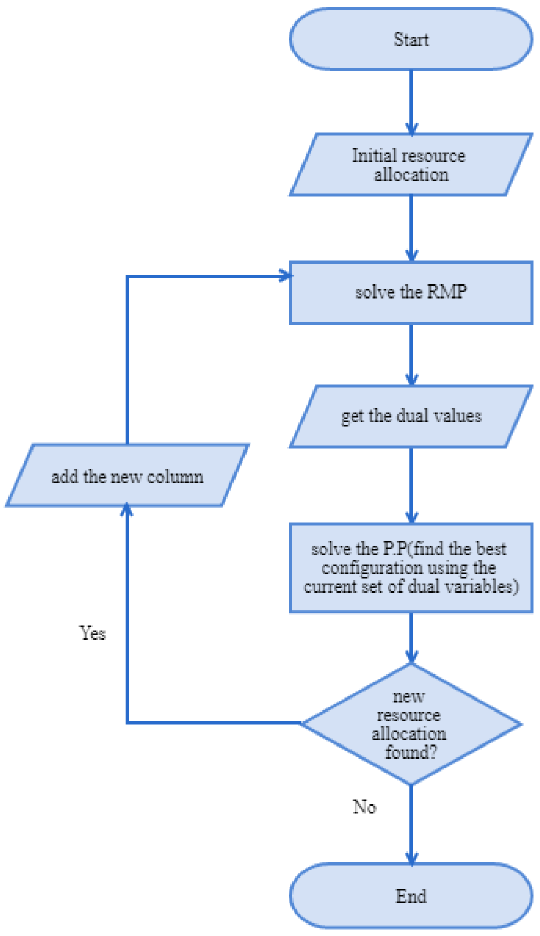

Start
Initial resource allocation
solve the RMP
get the dual values
solve the P.P(find the best configuration using the current set of dual variables)
new resource allocation found?
Yes
No
add the new column
End

FIGURE 4.1: Flowchart of the column generation algorithm

By using the definition of additive utilities and decomposing the utility of agents over their set of resources, we obtain

$$u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a})) = \sum_{r \in R} u^{\theta}_{ar}\, x_{ra}. \tag{4.11}$$

The expression of the reduced cost is written as follows:

$$\bar{c}_{\theta O} = c_{\theta O} - w^{\text{PROBA}}_{\theta} - \sum_{a \in A} \Bigg[\Big(w^{\text{IR}}_{a,\theta} + \sum_{\substack{\theta' \in \Theta \\ \theta' \neq \theta}} w^{\text{IC}}_{a,\theta,\theta'} \Big) \sum_{r \in R} u^{\theta}_{ar}\, x_{ra} \\ - \sum_{\substack{\theta' \in \Theta \\ \theta' \neq \theta}} w^{\text{IC}}_{\theta,\theta',\theta} \sum_{r \in R} u^{\theta'}_{ar}\, x_{ra} \Bigg] \tag{4.12}$$

where $w^{\text{PROBA}}_{\theta}$, $w^{\text{IR}}_{a,\theta}$, $w^{\text{IC}}_{a,\theta,\theta'}$ are the values of the dual variables associated with Constraints (4.7), (4.8), and (4.9), respectively, and where

$$c_{\theta O} = \sum_{a \in A} \sum_{O \in \mathcal{O}} u_a(\hat{\theta}_a, O(\theta_a, \theta_{-a})). \tag{4.13}$$

The pricing problem solves a resource allocation problem with respect to the reduced cost. Taking advantage of the fact that the pricing is in charge of generating the resource allocation, we will design pricing problems that correspond to different types of trades between the agents.

4.4.4.2 Many-to-One Transaction

We will first illustrate a model corresponding to the many-to-one types of trade. By assuming a_0 to be the agent initiator of the mechanism, agents can only exchange their resources through a_0. In this case the constraints of the pricing problem are then written as follows.

$$\sum_{a \in A} x_{ar} = 1 \qquad r \in R \tag{4.14}$$

$$x_{a'r} = 0 \qquad r \in R_a, a, a' \in A \setminus \{a_o\}, \tag{4.15}$$

$$x_{ar} \in \{0,1\} \qquad a \in A. \tag{4.16}$$

Indeed, if a resource initially belongs to $a \neq a_0$, it must first be exchanged with a_0, before it can be owned by a'. On the other hand, if it belongs to a_0, it can be exchanged with any agent $a \in A$. These are being satisfied in Constraints (4.15). Constraints (4.14) ensures that no resource can be owned by more than one agent.

4.4.4.3 One-to-One Transactions

One can restrict the set of interactions between the agents even more by only allowing one-to-one trades in which if agent a_1 trades a set of resources with agent a_2 then non of them is allowed to have trades with other agents in the system. In order to reach the following we need two more sets of variables:
$y_{aa'} = 1$ if there is at least one resource going from a to a', 0 otherwise
$y^r_{aa'} = 1$ if resource r moves from agent a to agent a', 0 otherwise.

Constraints of the pricing problem are then written as follows.

$$\sum_{a\in A} x_{ar} = 1 \qquad r \in R \tag{4.17}$$

$$x^{\text{init}}_{ar} + x_{a'r} - 1 \leq y^{r}_{aa'} \qquad a, a' \in A, r \in R \tag{4.18}$$

$$y^{r}_{aa'} \leq x^{\text{init}}_{ar},\ y^{r}_{aa'} \leq x_{a'r} \qquad a, a' \in A, r \in R \tag{4.19}$$

$$y^{r}_{aa} \leq y_{aa'} \qquad r \in R, a, a' \in A \tag{4.20}$$

$$\sum_{a'\in A} y_{aa'} \leq 1,\ \sum_{a'\in A} y_{a'a} \leq 1 \qquad a \in A \tag{4.21}$$

$$y_{aa'} + \sum_{a''\neq a} y_{a'a''} \leq 1 \qquad a, a' \in A \tag{4.22}$$

$$x_{ar} \in \{0,1\} \qquad a \in A, r \in R \tag{4.23}$$

$$y_{aa'} \in \{0,1\} \qquad a, a' \in A \tag{4.24}$$

$$y^{r}_{aa'} \in \{0,1\} \qquad r \in R, a, a' \in a. \tag{4.25}$$

Again, Constraints (4.17) enforce that each resource belongs to one agent. Constraints (4.18)-(4.20) define the possible conditions of a resource transfer from a to a', i.e., correspond to the linearization constraints of $y_{aa'} = x^{\text{init}}_{ar}\, x_{a'r}$, with $y^{r}_{aa'} = \max\limits_{r\in R} y^{r}_{aa'}$. Constraints (4.21) prevent a given agent a from sending to or receiving from more than one other agent, respectively. Constraints (4.22) force agent a to either receive or send (not both) from/to a single agent.

4.4.4.4 Many-to-Many Transactions

Many-to-many trades in which each agent is allowed to share any number of resources in its possession with any other agent in its neighbourhood is a sufficient condition to reach the optimal resource allocation (T.Sandholm [1998]). In this case the only constraint set of the pricing problem is limited to expressing the condition that a resource can be owned by only one agent at a time:

$$\sum_{a\in\mathcal{A}} x_{ra} = 1 \qquad r \in R. \tag{4.26}$$

Defining w_{ar} as follows:

$$w_{ra} = \left(w^{\text{IR}}_{a,\theta} + \sum_{\theta'\in\Theta\theta'\neq\theta} w^{\text{IC}}_{a,\theta,\theta'} \right) u^{\theta}_{ar} - \sum_{\theta'\in\Theta\theta'\neq\theta} w^{\text{IC}}_{\theta,\theta',\theta} u^{\theta'}_{ar}, \tag{4.27}$$

we can restate the pricing problem (PP$_{\theta O}$) as follows:

$$\max \sum_{a\in A}\sum_{r\in R} w_{ar}x_{ar} - w_{\theta}^{\text{PROBA}} \tag{4.28}$$

subject to:

$$\sum_{a\in A} x_{ar} = 1 \qquad r \in R \tag{4.29}$$

$$x_{ar} \in \{0,1\} \qquad a \in A, r \in R, \tag{4.30}$$

where:

$$w_{ar} = c_{\theta O} - w_a \times \alpha_{\theta O}^{\text{IR}} - w_{a,\theta',\theta}^{\text{IC}} \times \alpha^{\text{IC}}. \tag{4.31}$$

We then observe that, for many-to-many transactions, the pricing problem is a simple assignment problem, which can be solved in polynomial time. Indeed, we have the following theorem.

Theorem 4.1. *For many-to-many transactions, the pricing problem (outcome/type generator) can be solved in polynomial time, i.e., $O(n.|R|)$, where $n = |A|$. Therefore, the mechanism can be solved in polynomial time for the utilitarian social welfare objective.*

Proof. Observe that each pricing problem PP$_{\theta O}$ is decomposable, i.e., can be decomposed into $|A|$ independent sub-problems of the form PP$^{\,r}_{\theta O}$:

$$\max \sum_{a\in A} w_{ar}x_{ar} \tag{4.32}$$

subject to:

$$\sum_{a\in A} x_{ar} = 1 \tag{4.33}$$

$$x_{ar} \in \{0,1\} \qquad a \in A. \tag{4.34}$$

Each PP$^{\,r}_{\theta O}$ is polynomially solvable: its optimal value is $\max\limits_{a\in A} w_{ar}$ with $x_{\tilde{a}r} = 1$ for $\tilde{a} = \arg\max\limits_{a\in A} w_{ar}$ and all the others $x_{ar} = 0$. □

Any column generation decomposition model corresponds to a Dantzig-Wolfe model (Vanderbeck [2000]). While unusual, the column generation model was first studied, while its corresponding so-called compact formulation has not been looked at. In this section, we establish the compact formulation for the many to many type of transactions which was stated at section 4.4.4.4, in order to later investigate its scalability.

We define a new set p of variables such that p_{ar}^{θ} is equal to the probability of assigning resource r to agent a when type profile θ is reported. The compact formulation is stated below.

$$\max \sum_{a \in A} \sum_{\theta \in \Theta} \sum_{r \in R} u_{ar}^{\theta}\, p_{ar}^{\theta} \tag{4.35}$$

subject to:

$$\sum_{r \in R} u_{ar}^{\theta} p_{ar}^{\theta} \geq \sum_{r \in R_{\text{init}}^{a}} u_{ar}^{\theta} \qquad \theta \in \Theta, a \in A \tag{4.36}$$

$$\sum_{r \in R} u_{ar}^{\hat{\theta}} p_{ar}^{\hat{\theta}} \geq \sum_{r \in R} u_{ar}^{\theta} p_{ar}^{\theta} \qquad \theta, \hat{\theta} \in \Theta; a \in A \tag{4.37}$$

$$\sum_{a \in A} p_{ar}^{\theta} = 1 \qquad r \in R, \theta \in \Theta \tag{4.38}$$

$$p_{ar}^{\theta} \geq 0 \qquad \theta \in \Theta, a \in A, r \in R. \tag{4.39}$$

Constraints (4.36) express the individual rationality constraints. Here R_{init}^{a} is the set of the initial resources to each agent a, and constraints (4.37) ensure that the mechanism is incentive compatible using the Bayes-Nash equilibrium constraints.

Theorem 4.2. *Models* CG_AMD *and* COMPACT_AMD *are equivalent.*

Proof. Proof consists in re-interpreting Model CG_AMD as a so-called column generation model, and Model COMPACT_AMD as the compact model associated with CG_AMD, i.e., Model CG_AMD can be derived from COMPACT_AMD using a Dantzig-Wolfe decomposition. Reader who is not familiar with linear programming and decomposition techniques in mathematical programming is referred to (Lasdon [1970]).

Dantzig-Wolfe decomposition relies on the Minkowski theorem, which states that all bounded polyhedra P are convex combinations of their extreme points $\text{EXTR}(P)$, i.e., any point x of P can be written $x = \sum_{e \in \text{EXTR}(P)} \lambda_e e$ with $\sum_{e \in \text{EXTR}(P)} \lambda_e = 1$ and $\lambda_e \geq 0$.

As the polyhedron $P^{\text{COMPACT\_AMD}}$ defined by the set of constraints (4.36)-(4.39) of Model COMPACT_AMD is bounded thanks to constraints (4.38), we now use Minkowski's theorem for polytopes (i.e., bounded polyhedrons). Let $E = \{e_j^{ar\theta} : j \in J\}$ be the set of extreme points of $P^{\text{COMPACT\_AMD}}$, where J is the index set of the extreme points. Then, for any $(a, r, \theta) \in A \times R \times \Theta$, there exists a set of scalars (λ_j^{θ}) such that we can rewrite each p_{ar}^{θ} in the following way:

$$p_{ar}^{\theta} = \sum_{j \in J} \lambda_j^{\theta} e_j^{ar\theta} \qquad a \in A, r \in R, \theta \in \Theta, \tag{4.40}$$

with $\sum_{a \in A} e_j^{ar\theta} = 1$ for $r \in R, j \in J, \theta \in \Theta$. After substituting p_{ar}^{θ} by the convex combination of its extreme points using 4.40 in constraints (4.36)-(4.39), and after re-interpreting each outcome

O as an the extreme points e_j, we can then establish a one-to-one correspondence between (4.36)-(4.39) and (4.7)-(4.9).

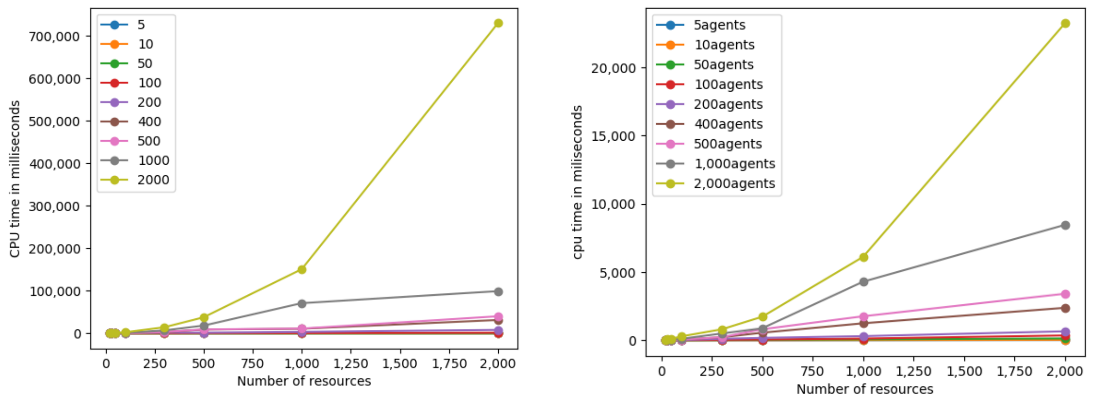
For example, consider Constraints (4.38). Using the decomposition into extreme points for given r and θ, Constraints (4.38) can be rewritten:

$$\sum_{a\in A} p^{\theta}_{ar} = \sum_{a\in A}\left(\sum_{j\in J}\lambda^{\theta}_j e^{ar\theta}_j\right) = \sum_{j\in J}\underbrace{\left(\sum_{a\in A} e^{ar\theta}_j\right)}_{\text{equal to 1}}\lambda^{\theta}_j = \underbrace{\sum_{j\in J}\lambda^{\theta}_j}_{\sum_{O\in\mathcal{O}} g^{O}_{\theta}} = 1 \quad (4.41)$$

which is equivalent to 4.7. Similar calculations allow the derivation of the other constraints of Model COMPACT_AMD from the constraints of Model CG_AMD. □

However, this new formulation of the polyhedron P now has as many variables as the number of extreme points of P. The reader is referred to, e.g., (Vanderbeck [2000]) if not familiar with Dantzig-Wolfe decomposition.

The interest of the COMPACT_AMD Model (given by (4.35)-(4.39)) lies in its polynomial number of variables, in contrast to the exponential number of variables of the CG_AMD Model (given by (4.7)-(4.10)). Both models correspond to linear programs.



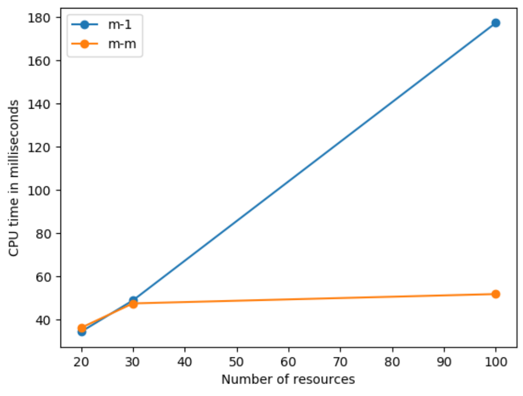
(a) CG_AMD Model

(b) COMPACT_AMD Model

FIGURE 4.2: CG_AMD CPU time vs. COMPACT_AMD

4.5 Experimental Results

We implemented all the models and algorithms discussed in the previous sections, and used Cplex (Cplex [2014]) to solve the linear (integer) programs. We discuss the generation of the data sets in Section 4.5.1. And then conduct a comparative analysis of the different trades, using the different four proposed models, see Sections 4.5.2 and 4.5.3.


(a) M-M vs M-1

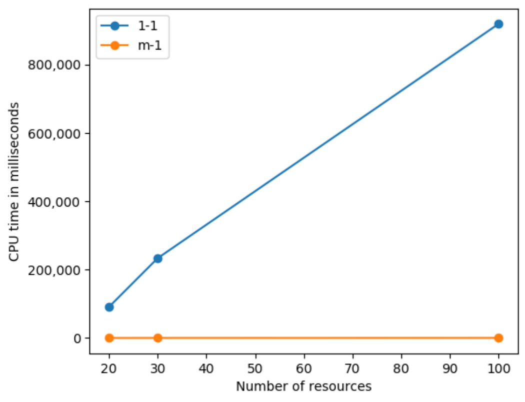
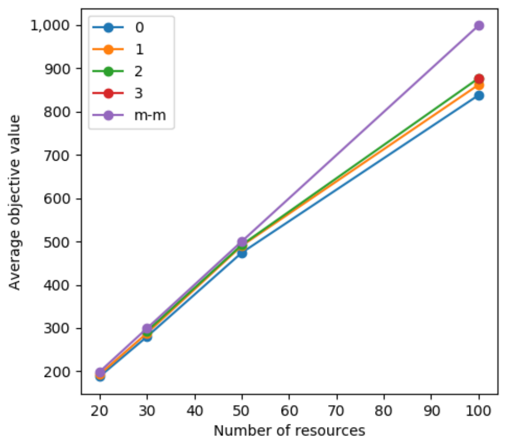

(b) M-1 vs 1-1

FIGURE 4.3: Comparison of different types of trades - 50 agents


(a) Selecting the column with the highest probability

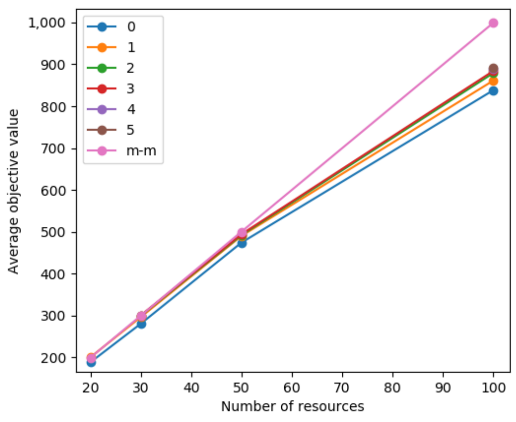

(b) selecting the column with the highest SW

FIGURE 4.4: Overcoming the local optimality having multiple rounds having 50 agents

4.5.1 Data sets

Data sets were randomly generated for a given number of agents and resources. As in the literature, we consider only one type per agent. The preference/type of each agent towards a resource is generated at random in $\{1, 2, \ldots, 10\}$. For each data set, the initial resource allocation has been computed using Cplex (Cplex [2014]) by solving a resource allocation problem. For each number of agents and resources, we generated 10 data-set instances and plot Figures 4.2, 4.3 and 4.5 with the average taken over them.

4.5.2 Column Generation vs. the Compact Formulation

Both CG_AMD and COMPACT_AMD were solved on data-sets with an increasing number of agents starting from 5 agents, reaching to 2,000 as well as an increasing number of resources

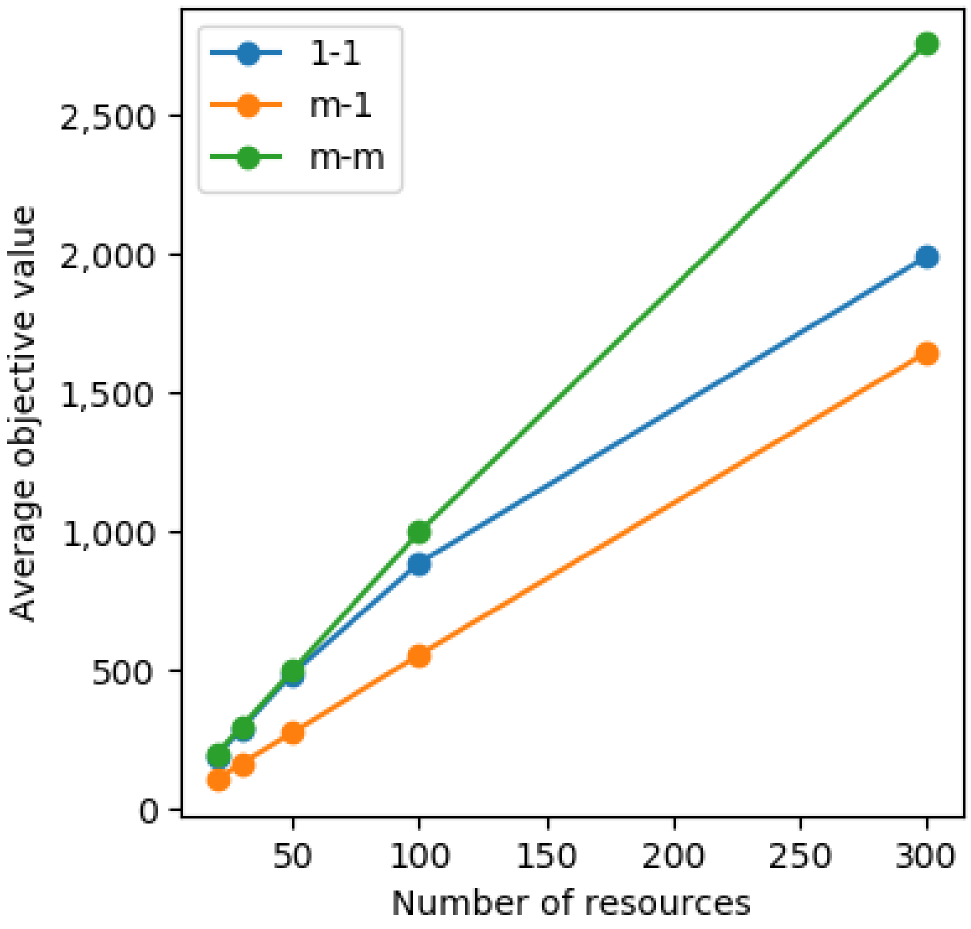


FIGURE 4.5: Comparison of the objective values for the case of having 50 agents in the system

varying from 20 to 2,000.

In Figure 4.2, the horizontal axis denotes the number of resources and the vertical axis indicates the average CPU time in mili-seconds taken over the ten different data sets associated with each (number of agents,number of resources) pair.

As one can observe, the COMPACT_AMD model beasts the CG_AMD model in terms of CPU time and is faster despite the fact that the pricing problem of the CG_AMD runs in polynomial time.

The CG_AMD model however, provides us with more versatility and can be used to solve the AMD problem with respect to different types of trades, i.e., many-to-one trades or one-to-one trades for data-sets significantly larger than the ones existing in the literature which will be discussed in the sequel.

4.5.3 Comparison of the Different Column Generation Models

By using the Column Generation algorithm to solve the models corresponding to different types of trades presented in Sections 4.4.4.3, 4.4.4.2 and 4.4.4.4, we could compare the models with respect to their objective value or the CPU time that the Column Generation algorithm requires to reach the final resource allocation of the problem .

The value of the social welfare is influenced not only by the neighbourhood of each agent but also by the negotiation protocols/trades that are permitted in the social network. In this work, we considered the social graph of the network to be a complete graph and have analyzed the effect of having different negotiation protocols (types of trades) in the system.

We will first present a comparative study of our Column Generation models in terms of their computational time and then will continue the comparison in terms of their objective values. We will finish this section by addressing the local optimality issues that occur for the many-to-one and one-to-one cases and provide solutions to improve the objective value of the mechanism.

4.5.3.1 Comparison of the CPU times

We have compared the models with respect to the CPU time that the column generation algorithm needs for each one of them to reach an equilibrium by taking the average over the ten data sets generated randomly for each (number of agents,number of resources) tuple. As one can observe in Figure 4.3, for a fix number of agents (in this case 50), the many to many model for which the pricing problem can be solved in polynomial time, will surpass the other column generation models as the number of resources grows. See Figures 4.3(a), 4.3(b).

By taking advantage of the flexibility of the Column Generation algorithm, we solved the AMD problem for large data instances for the many-to-many case and for the many-to-one types of trade for up to 1,000 agents and 1,000 resources and the one-to-one instances for up to 50 agents and 300 resources. Indeed, as shown in Figure 4.3(b), the one-to-one type of trade takes significantly more time to converge than the other two cases. For these experiments the number of agents was fixed to 50 and the number of resources varied from 20 to 100.

The many-to-many model is highly scalable regardless of if we are solving it using the COMPACT_AMD or the CG_AMD, i.e., it can be used to solve the problem for data instances up to 2,000 agents and 2,000 resources in seconds. The many-to-one model eventually runs slower than the many-to-many model as the number of the agents or resources increases. The one-to-one model runs slowest and faces scalibility issues much faster than the other two models.

4.5.3.2 Comparison of the Objective Value

The results we have obtained using both the COMPACT_AMD or CG_AMD are aligned with what was previously shown in (T.Sandholm [1998]). That is: Using many-to-many trades (the so called OCSM-contracts) in a mechanism that satisfies the individual rationality constraints, one can reach the optimal resource allocation despite the agents being self-interested. However, setting limits on the types of the trades will indeed affect the social welfare of the mechanism.

In Figure 4.5, the vertical axis denotes the average objective value taken over the ten different data instances corresponding to each (number of agent, number of resources) pair and the horizontal axis shows the number of resources. In this figure, we are comparing the objective value reached by running each of the models presented in Section 4.4.4.1 for a fixed number of agents,in this case 50 agents.

As one can observe, by allowing many-to-many trades, we are reaching the highest social welfare by acquiring the global optimum. Moreover, the objective value obtained when solving the problem by allowing only one-to-one trades is always greater than or equal to the objective value of the mechanism when only many-to-one trades are allowed. these results hold for any number of agents and resources in our test cases.

4.5.3.3 Local Optimality

By allowing either only one-to-one or only one-to-many trades amongst the agents, the mechanism usually gets stuck in the local optimum due to the agents being self-interested and not being omniscient.

In the case of having only many-to-one trades, the mechanism usually gets stuck in a local optimum due to the structure of this type of trade. Indeed, agents are not allowed to negotiate among themselves and have to use the agent initiator as a mediator to their trades even though an exchange among themselves might grant each one of them a higher utility. A self-interested and rational agent initiator could easily take advantage of its position in the system and keep its most favorite resources to it-self even though giving those items away to other agents would result in a higher social welfare in the mechanism. This might cause the mechanism to get stuck in a local optimum and shows why a centralized approach would not always lead to the optimal resource allocation in the AMD problem.

Using the many-to-many or one-to-one types of trades we will overcome the issue with a self-interested central entity. Although using many-to-many trades will result in reaching the global optimum, one-to-one trades will still be trapped in a local optimum as a result of the agents not being omniscient. Indeed, the mechanism is very sensitive towards the initial resource allocation regardless of when dealing with one-to-one trades or many-to one trades. This happens because most of the time the agent having more resources has more power and flexibility towards accepting or declining a trade. Knowing this, We have designed an algorithm to overcome the issue of local optimality for one-to-one transactions.

In order to overcome the local optimality issue of the one-to-one model, we took an iterative approach towards solving it. Meaning that we allow the agents to distribute their resources via one-to-one trades among themselves in multiple rounds. In each round, after the termination of the column generation algorithm, we selected the column which has provided us with the highest social welfare and fed the resource allocation associated with that column to the model as our new initial resource allocation and ran the problem for another round. The following process continued until either we exceeded a certain number of rounds or could not improve the objective value since the previous round more than a certain epsilon.

Other algorithms and heuristics can also be considered to escape local optimality. For example, at the each round instead of selecting the resource allocation corresponding to the highest social welfare one can select the resource allocation associated with the column that has the highest probability, i.e., g^O_θ and repeat the previous process.

We have conducted experiments using both of the suggested techniques. However, the results we have obtained by taking the resource allocation corresponding to the column with the highest utility value were most of the times better than the ones of taking the column with the highest g^O_θ. This means that even though the algorithm will go through more iterations when selecting the column with the highest social welfare, it will eventually terminate with a objective value closer to the optimum social welfare than when selecting the highest probability g^O_θ.

In Figure 4.4, we analyze the growth of the objective value in each of the rounds for data-sets with a fixed number of agents, i.e., 50 agents and with the number of resources varying from 20 to 100. In order to see the growth in each iteration and compare the final result with the optimal social welfare obtained by solving the many-to-many model we are **not** taking the average over our data sets for these experiments. Here, the horizontal axis indicates the number of resources and the vertical axis represents the objective value of the mechanism. With each color representing a different round of the heuristic algorithm.

When solving the AMD problem with one-to-one types of trades, we use a multiple round algorithm, repeating the solution of the AMD problem until there was no improvement in the solution generated in the previous round. We observe that the objective value gets closer to the global optimum before reaching the number of round threshold. At the beginning of each round, the initial resource distribution is the one with the largest social welfare output in the previous round.

Each line in the figure corresponds to the change in the objective value in each round based on the number of resources.

Moreover, we have also solved the many-to-many model for the same data sets as the one-to-one model with multiple rounds. We then drew the line corresponding to it in Figures 4.4(b) and 4.4(a) in order to be able to see how close we got to the global optimum by running the one-to-one algorithm in multiple rounds.

As it can be observed, although the social welfare reached by the one-to-one mechanism is increasing after each round, it is always less than or equal to the objective value obtained by running the many-to-many model.

4.6 Conclusion

Using a large scale optimization algorithm, namely the column generation technique, we have very significantly reduced the scalability issues of the solution of the Automated Mechanism Design (AMD) problem. By taking advantage of the versatility of the pricing problem of the column generation technique, we solved the AMD problem for the multi-agent resource allocation problem subject to different types of trades.

When allowing one agent to be the agent initiator and to play the role of a central entity, i.e., many-to-one trade, the mechanism usually gets stuck in a local optimum due to the self-interested behavior of the agent initiator. That last issue can be alleviated by considering a more distributed approach. Limitation could be set on the types of trades among the agents from only allowing two agents to trade their goods at a time, i.e., one-to-one/bilateral trades to a multilateral/many-to-many approach for which all agents are allowed to exchange any number of resources in their resource bundles among themselves. The many-to-many trades will always converge to an optimal resource allocation, contrarily to what is happening with the one-to-one approach.

Moreover, we introduced the compact formulation of Automated Mechanism Design problem for them many-to-many types of trade when no money is allowed in the system by showing that the many-to-many column generation model is a Dantzig-Wolfe decomposition of that formulation. The many-to-many model is more encountered in real world applications and has proven to provide us with a tool to reach the optimal resource allocation in the system. More importantly, the compact formulation of the problem which has a polynomial number of variables surpasses the column generation algorithm in terms of computational time and reaches the optimal resource allocation much faster.

Using the algorithms and models presented in this paper we could solve the AMD problem for much larger data-sets than the ones found in the literature. We have managed to solve the problem by allowing multilateral trades for up to 2,000 agents and 2,000 resources in less than 25 seconds. The other models corresponding to the other types of trades have also been used to solve large data sets, i.e., up to 1,000 agents and 1,000 resources for the many-to-one transactions and 50 agents and 300 resources for the one-to-one transactions.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we studied the Automated Mechanism Design (AMD) problem. While there are many studies on AMD, there are very few scalable algorithms, i.e., algorithms that can solve instances with more than a few hundred agents and resources in a few seconds. The first key contribution of this thesis is to move the scalability of the algorithms to a new level, i.e., few seconds for multi-agent instances with up to 1,000 agents, making the practical use of AMD much easier.

We designed four new optimization models to solve the AMD problem, subject to different types of trades. We investigated the effect of having different negotiation protocols on the social welfare and the final resource allocation of the mechanism when each of the agents reports only one type towards each of the resources.

The social welfare of the mechanism is very sensitive towards the initial resource allocation of the mechanism and the different types of trades that are allowed among the agents. Indeed, different negotiation protocols will affect the outcome of the mechanism in many ways such as the final resource allocation, the social welfare or the run time of the problem. Most of the work in the literature is focused on bilateral (one-to-one) trades, i.e., trades that only allow two agents to trade their resources between themselves at a time. Even though many-to-many (multilateral) trades are much more common in, e.g., social networks, not many scalable algorithms were developed to solve the AMD problem with respect to these trades. All proposed models and algorithms in this thesis can be used to solve the AMD problem for much larger data sets than the ones existing in the literature, mainly up to 2,000 agents and 2,000 resources for the very common **multilateral** types of trade as well as large data sets corresponding to the other types of trades, e.g., 1,000 agents and 1,000 resources for the many-to-one trades. In both cases, solution can be obtained in few seconds.

After designing the models and efficient algorithms to solve them, we investigated the following questions:

- Is it always the case that by solving the AMD problem using each of the models, we could reach the optimal resource allocation? If not, why?
- If the mechanism gets stuck in a local optimum, what could be done?
- How fast and efficient are our algorithms compared to each other and compared to the existing work in the literature?

We have shown that not all of the models would guarantee reaching an optimal resource allocation. Indeed, the self-interested behavior of the agents plays a major role with this regard and could actually result in the mechanism getting stuck in a local optimum solution. This can be noticed when solving the problem using a centralized approach by applying our many-to-one model. In that case, the agent initiator, which is a self-interested and rational entity, took advantage of its power. Indeed, as the main distributor of the resources, the agent initiator kept those resources that would lead to an increase in its utility, even though by giving them away to other agents a higher social welfare could be achieved.

The issue was fixed by taking a distributed approach towards the problem using our one-to-one model. In the so-called one-to-one model, agents can negotiate their initial set of resource with another without the need of a central entity under the condition that only two agents are allowed to trade their resources at a time. Using the one-to-one model, even though we were still far from the optimal resource allocation, we reached a higher social welfare than the one reached via the many-to-one approach. Moreover, we have designed a heuristic algorithm that would help us escape the local optimum and improve the social welfare by allowing multiple rounds of bilateral trades.

Unfortunately, despite having a heuristic for the bilateral trades, we did not always reach the global optimum. The main reason for this is that the agents are not omniscient. The following issue was solved by introducing the many-to-many models which correspond to multilateral trades and will reach the optimal resource allocation.

Although most of the times the objective value obtained by using the one-to-one model is much better than the one obtained using the many-to-one model, the many-to-one algorithm runs much faster. Moreover the second many-to-many model which can be solved in polynomial time and is the compact formulation of the column generation many-to-many model can be used to solve data sets of the size of 2,000 agents and 2,000 resources in less than 25 seconds.

5.2 Future Work

Future work will include solving the AMD problem for when each of the agents can report multiple types (e.g., private information) towards the resources. It is very common in practice for an agent to report multiple types towards a resource or a resource bundle. This mainly happens because of the self-interested behavior of the agents. Although an agent has a certain value/preference towards a resource, the value/preference it reports for a resource in the mechanism depends on the values/preferences reported by the other agents for which it is usually assumed that the agent has some partial information about. For example, this information could be the probability distribution of the types of the other agents. Solving the problem when having multiple types per agent is of high interest in the industry and has numerous applications such as the sponsor search auction which is being used by Google. However, it is not easy to generate multiple type profiles associated with each agent as it involves using advanced tools in microeconomics and modeling the partial information of other agents. Most of the times it is even difficult for an agent herself to know her truthful types (Parkes [2001]) and generating multiple types would be even a more difficult quest.

We also plan to test our models and algorithms on a social network that is not a complete graph, as in the numerical experiments of Chapter 4 in this thesis. Indeed, we plan to use some real world social networks available in, e.g., (J.Leskovec and A.Krevl [2014]) and (R.A.Rossi and N.K.Ahmed [2015]) with at least up to 8,000 nodes/agents. Our expectations are that even though in the context of a complete graph, the COMPACT_AMD model is more efficient than the CG_AMD model from a computational time perspective, when moving to a more social graphs with a weaker density, the column generation formulation, i.e , the CG_AMD model will outperform the compact formulation (COMPACT_AMD) in terms of computation time and, consequently, would be more scalable.

We will also work on designing mechanisms for when money is allowed in the system by implementing the Vickrey-Clarke-Groves (VCG) mechanisms. In a lot of application domains, the use of money is not prohibited and that would add some more complexity to the problem as one has to design payment functions that would force the agents to report their truthful preferences towards the resources without violating the individual rationality of the mechanism. This is not a trivial task and can be achieved using the VCG mechanisms.

Finally, we will study the cases for which the utility functions are not modular. As a matter of fact when bundling is allowed in the mechanism, it is difficult to set the price of a bundle in a way to encourage people to buy the bundle of resources such that the revenue of the seller is maximized. The problem is usually modeled by using different utility functions such as supermodular or sub-modular utility functions. The following is a hot and ongoing research topic in the world of mechanism design and has numerous applications such as Amazon‘s Amazon

Prime system. Other application contexts are when designing a fair or envy-free mechanism which is strongly affected by the type of the utility functions of the agents.

Bibliography

A. Mas-Colell and M.D. Whinston and J.R. Green. *Microeconomic Theory*. Oxford University Press, USA, 1995.

M.R. Andersson and T.W. Sandholm. Leveled commitment contracts with myopic and strategic agents. In *Conference on Artificial Intelligence (AAAI)*, pages 1–7, USA, California, 1998.

K.J. Arrow. *Social Choice and Individual Values*. Yale University Press, New Haven, USA, 1963.

F. Asselin, B. Jaumard, and A. Nongaillard. A technique for large automated mechanism design problems. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology (IAT)*, pages 467–473, Hong Kong, China, December 18-22 2006.

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Negotiating over small bundles of resources. In *AAMAS'05 - Autonomous Agents and Multiagent Systems*, pages 296–302, EU, The Netherlands, Utrecht, July 25-29 2005. ACM Press.

Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Reaching envy-free states in distributed negotiation settings. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1239–1244, India, Hyderabad, January 2007.

Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent resource allocation with k-additive utility functions. *Annals of Operations Research*, 163:49 – 62, October 2008.

Y. Chevaleyre, U. Endriss, and N. Maudet. Distributed fair allocation of indivisible goods. *Artificial Intelligence*, 242:1–22, January 2017.

Y. Chevaleyre *et al.* Issues in Multiagent Resource Allocation. *Informatica*, 30:3–31, 2006.

V. Chvatal. *Linear Programming*. Freeman, USA, 1983.

V. Conitzer and T. Sandholm. Complexity of Mechanism Design. In *Uncertainty in Artificial Intelligence (UAI)*, pages 103–110, San Francisco, 2002. Morgan Kaufmann.

V. Conitzer and T. Sandholm. An algorithm for automatically designing deterministic mechanisms without payments. In *Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, volume 1, pages 128–135, New York, NY, USA, 2004.

Cplex. *IBM ILOG CPLEX 12.6 Optimization Studio*. IBM, 2014.

R. Dash, D. Parkes, and N. Jennings. Computational mechanism design: A call to arms. *IEE Intelligent Systems*, 18:40–47, 2003.

M. de Weerdt, Y. Zhang, and T. Klos. Multiagent task allocation in social networks. *Autonomous Agents and Multi Agent Systems*, 25:46 – 86, 2012.

Ulrich Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Negotiating socially optimal allocations of resources. *Journal of artificial intelligence research*, 2006.

S. Estivie, Y. Chevaleyre, U. Endriss, and N. Maudet. How equitable is rational negotiation? In *AAMAS'06 - Autonomous Agents and Multiagent Systems*, pages 866–873, Japan, Hakodate, May 8-12 2006. ACM Press.

D. Friedman and J. Rust. *The Double Auction Market: Institutions, Theories and Evidence*. Addison-Wesley, USA, 1993.

J.Leskovec and A.Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

L.G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

L.S. Lasdon. *Optimization Theory for Large Systems*. MacMillan, New York, 1970.

M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.

H. Moulin. *Axioms of cooperative decision making*. Cambridge University Press, England, 1988.

M.R.Andersson and T.Sandholm. Contract types for satisficing task allocation: Ii texperimental results. In *AAAI Spring Symposium: Satisficing Models*, pages 68–75, USA, 1998.

H. Narasimhan, S. Agarwal, and D.C. Parkes. Automated mechanism design without money via machine learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 433 – 439, New York City, US, 2016.

A. Nongaillard, P. Mathieu, and B. Jaumard. A multi-agent resource negotiation for the utilitarian social welfare. In *9th Annual International Workshop on Engineering Societies in the Agents World (ESAW)*, pages 208 –226, Berlin, 2008. Springer-Verlag.

S. Park and S.-B. Yang. An efficient multilateral negotiation system for pervasive computing environments. *Engineering Applications of Artificial Intelligence*, 21:633–643, 2007.

D.C. Parkes. *Classic Mechanism Design*. PhD thesis, University of Pennsylvania, 2001.

R.A.Rossi and N.K.Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. URL `http://networkrepository.com`.

T. Sandholm. Automated Mechanism Design: A New Application Area for Search Algorithms. In *9th International Conference on Principles and Practice of Constraint Programming*, volume 2833 of *Lecture Notes in Computer Science*, pages 19–36, Berlin, Heidelberg, 2003. Springer.

T.W. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

T.Sandholm. Contract types for satisficing task allocation: I theoretical results. In *AAAI Spring Symposium: Satisficing Models*, pages 68–75, USA, 1998.

F. Vanderbeck. On integer programming decomposition and ways to enforce integrality in the master. *Operations Research*, 48(1):111 – 128, Jan. - Feb. 2000.

D. Ye, M. Zhang, and A.V. Vasilakos. A survey of self-organization mechanisms in multiagent systems. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 47:441 – 461, 2017.