MATHEMATICAL PROGRAMMING APPROACH TO MULTIDIMENSIONAL MECHANISM DESIGN FOR SINGLE MACHINE SCHEDULING

J. Duives

APPLIED MATHEMATICS
DISCRETE MATHEMATICS AND MATHEMATICAL PROGRAMMING

EXAMINATION COMMITTEE
Prof. Dr. M.J. Uetz
Dr. J.B. Vink-Timmer
Prof. Dr. J.L. Hurink

UNIVERSITY OF TWENTE.

08-17-2011

Preface

This master thesis is the result of my graduation project at the "Discrete Mathematics and Mathematical Programming (DMMP)" group of the University of Twente. Using this opportunity, I would like to thank several people for their assistance, guidance and amusement during this project.

First of all many thanks go to Marc Uetz, my supervisor at the University of Twente. After a smooth cooperation during my internship, I happily accepted a graduation project under your "service". Many times we have discussed and analysed problems we encountered for hours, possibly more than you intended to. I am grateful for this.

Furthermore I would like to thank all people who have made my graduation project that more fun and easy going. For example, I really enjoyed the sometimes profound discussions with fellow students and professors during coffee breaks an during lunch. Moreover, I could always share my joy or disappointments with housemates, but also with friends from badminton or elsewhere.

Finally I would like to thank my family and girlfriend for their overall support. They helped and kept me motivated not only during my graduation project, but also during the study preceding this thesis.

J. Duives.
Enschede, August 19, 2011.

Contents

1 Introduction

Mathematical optimization can be used to solve many economic problems concerning logistics and production. Collecting all information relevant for the problem, one can 'simply' select a solution optimizing a certain global objective from some set of available alternatives. However, quoting Kantorovich [7], many of these classical optimization problems do not relate to realistic situations: "I want to emphasize again that the greater part of the problems of which I shall speak, relating to the organization and planning of production, are connected specially with the Soviet system of economy and in the majority of cases do not arise in the economy of a capitalist society". Even more, "There [capitalism] the choice of output is determined not by the plan but by the interests and profits of individual capitalists". Kantorovich is emphasising the fact that in capitalist economics, solving economic problems where certain decisions are left to individuals instead of a central authority, using classical optimization, is useless. For example, people may have a personal objective that induces a preference that does not match with the performance of the system as a whole. In those cases people might, based on their own and other peoples preferences, act strategically in order to manipulate the decision made by the central authority. Due to this strategic behaviour of individuals, classical optimization fails. The mathematical models to analyse such strategic situations are studied in Game Theory [12, 11].

A special class of games in Game Theory are games with incomplete information. Whereas otherwise, the preferences of individuals are assumed to be known to other individuals as well as to the central authority, in games of incomplete information the actual preferences of people are private information. In this type of games, (additional) manipulation of people may occur by reporting a false preference. An additional complication in comparison to 'normal' games, is that individuals do not know which decisions are beneficial for other people, as this depends on those peoples' preferences. To be able to optimize a given objective for games with incomplete information we need a different optimization technique. In 2007 the Nobel prize in Economics was awarded to Leonid Hurwicz, Eric Maskin, and Roger Myerson for having laid the foundations of mechanism design theory [1]. Quoting Sandholm [13]: "Mechanism design is the art of designing the rules of the game such that a socially desirable outcome is reached despite the fact that each agent acts in its own self-interest". In other words, mechanism design helps to optimize some global objective, accounting for the fact that individuals might, based on private preferences, act strategically.

In this thesis we consider a classical single machine scheduling problem. Given is a set of risk-neutral jobs that need to be processed non-preemptively on a single machine, that can handle only one job at a time. These jobs act selfishly as they all have a personal objective: to be processed as soon as possible. Each job has a processing time p_j and a disutility w_j for waiting one unit of time, which both can be private information. We refer to the private information of a job as its type. Although jobs do not known the actual type of other jobs, we assume that they do share common beliefs about other jobs' types in terms of (discrete) probability distributions. An allocation rule, taking the role of central authority, assigns to each (possibly untruthful) report of jobtypes, a schedule σ, denoting the order in which the jobs are processed on the machine. We assume that jobs' preferences over possible schedules are expressed as $-w_j S_j(\sigma)$, where $S_j(\sigma)$ is the start time of job j in schedule σ.

Depending on their disutility for waiting, jobs are compensated for waiting in the form of a payment. In this setting a mechanisms consists of an allocation rule and a payment scheme, defining the payments jobs receive to be compensated for waiting. The payments influence the objectives of jobs as follows. Let us denote the utility of job j when schedule σ is chosen and it receives payment π_j, by $\pi_j - w_j S_j(\sigma)$, referred to in the literature as quasi-linear utility with respect to payment π_j [9][1]. We assume that jobs seek to optimize their (expected) utility. We only consider direct revelation mechanisms, that is, mechanisms in which the only decision made by jobs, is the report of their type. Even more, by making use of Myersons' famous revelation principle [10], we may restrict ourselves to truthful or incentive compatible mechanisms, which are mechanisms where jobs have the incentive to report their type truthfully. More specific, Bayes-Nash incentive compatible (BNIC) mechanisms motivate jobs to report truthfully, provided that other jobs also do so, whereas (stronger) dominant strategy incentive compatible (DSIC) mechanisms motivate jobs to report truthfully regardless of what other jobs do. In this setting our goal is to find mechanisms that minimize the expected total payment made to the jobs, while motivating jobs to report their weight truthfully (either BNIC or DSIC).

This problem, mainly the special case of 1-dimensional types (public processing times p_j and private w_j), has been considered earlier in a paper by Heydenreich et al. [5]. They prove that serving jobs in order of non-increasing ratio of modified weights over processing times, $\bar{w}_j/p_j$, is optimal

[1]Note that for a given schedule σ, the utility is also linear in w_j, i.e. jobs have linear utility.

[10], i.e. minimizes the payments made to the jobs while being Bayes-Nash incentive compatible. Moreover, Heydenreich et al. [6] prove that Bayes-Nash incentive compatibility and dominant strategy incentive compatibility is equivalent in the sense that if there exists a mechanism that is Bayes-Nash incentive compatible, then there exists a dominant strategy incentive compatible with the same expected total payment. Finally, in search for a closed formula for the optimal mechanism also in the 2-dimensional setting (both p_j and w_j private), Heydenreich et al. [5] propose an example to show that optimal mechanisms in the 2-dimensional setting in general do not satisfy a condition called IIA, independence of irrelevant alternatives. This example gives a hind towards intractability of the 2-dimensional mechanism design problem. However, that example was flawed.

Motivated by the questions left open in [5], in this thesis we are interested in getting more insight into properties of (optimal) mechanisms for the 2-dimensional setting. In particular, we are interested in the IIA condition, the minimal condition that an optimal mechanism should have if a closed formula were to exist. Constructing a new example by hand to prove that optimal mechanisms in the 2-dimensional setting in general do not satisfy IIA, turned out to be difficult and time consuming. Therefore we decided to switch to a more systematic approach, i.e. optimal mechanism design by mathematical programming, also known as automated mechanism design [2, 13]. In automated mechanism design the mechanism is designed 'automatically' for the setting and objective at hand, where automatically refers to the use of IP solvers.

In the flavour of recent work on automated mechanism design as proposed by Conitzer and Sandholm [2, 13], we formulate the optimal mechanism design problem for this scheduling application as Mixed Integer Linear Programming Problem (MIP). This MIP formulation allows us, using ILOG CPLEX as solver for the MIPs, to compare optimal solutions for different types of mechanisms in the scheduling problem. Indeed, by this approach we are able to reconfirm that optimal mechanisms in the 2-dimensional setting in general do not satisfy IIA, reconfirming a theorem formulated in [5]. Additionally we use this MIP to prove that for the 2-dimensional setting, BNIC and DSIC are in general not equivalent in the sense that there is an instance where the minimal expected total payments achieved by the DSIC mechanism exceed those of the optimal Bayes-Nash mechanism. Besides these general results, we compare different types of mechanisms in specific types of instances to possibly strengthen our findings.

The organisation of this thesis is as follows. In Section 2 first we dis-

cuss the well known theory for the non-strategic single machine scheduling problem, after which we give a detailed sketch of the strategic problem we consider in the remainder of this thesis. Also in this section, we discuss the results Heydenreich et al. found for the 1-dimensional setting of the problem as well as some related theorems by Manelli and Vincent [8] and Gershkov et al. [4]. In Section 3 we propose a MIP formulation for the optimal mechanism design problem, for both BNIC and DSIC mechanisms, which is the most important part of our solution method. A complete description of our solution method, including implementation details, is discussed in Section 4 whereas the re results of our research can be found in Section 5. Finally we conclude this thesis with a summary of our results and some recommendations for future research.

2 Optimal Mechanisms for Scheduling

In this section we start by discussing the non-strategic version of the single machine scheduling problem. Then we switch to the strategic single machine scheduling problem. First we discuss the 1-dimensional setting for this problem as well as some results by Heydenreich et al [5]. Even more, we elaborate on some related results by Manelli and Vincent [8] and Gershkov et al. [4]. Eventually we switch to the 2-dimensional single machine scheduling problem together with the flawed counterexample that formed the starting point for our research.

2.1 Single Machine Scheduling Problem

Let us consider the standard single machine scheduling problem. Given is a set of jobs $J = \{1, \ldots, n\}$, which have to be processed non-preemptively on a single machine that can handle one job at a time. Each job j has a processing time p_j and a disutility for waiting one unit of time, also called its weight w_j, both publicly known. Let $\mathfrak{S} = \{\sigma | \sigma \text{ is a permutation of } (1,\ldots,n)\}$ be the set of feasible schedules, i.e. the order in which jobs are processed on the machine. Denoting by σ_j the position of job j in schedule σ, the start or waiting time of job j is represented by $S_j(\sigma) = \sum_{\sigma_k < \sigma_j} p_k$. Note that we do not assume idle time, i.e. jobs are processed immediately after one another.

In this setting, all decisions are made by a central authority, e.g. scheduler, who chooses an order in which to process the jobs and we do not need to account for strategic behaviour of jobs. One of the standard objectives for this problem is to minimize the sum of weighted completion times, or equivalently, minimize the sum of weighted start times. Note that the latter is identical to the total disutility for waiting. This standard objective is optimized by a well known list scheduling algorithm known as Smith's rule [14], i.e. scheduling jobs in order of non-increasing ratio of weight over processing time w_j/p_j. From this standard single machine scheduling problem we switch to the strategic version of this problem, which we consider in the remainder of this thesis.

2.2 The 1-Dimensional Setting

A first departure from the non-strategic setting is that in the strategic setting we have a set of selfish jobs that act strategically. For the 1-dimensional setting we assume that p_j, the processing time of job j, is still publicly known,

whereas w_j, the weight of job j, is private information[2]. Although the weight of a job is private information, other jobs share common beliefs about jobs' types in terms of probability distributions. Let $W_j = \{w_j^1, \ldots, w_j^{m_j}\}$ denote the set of possible weights of job j. The corresponding (finite discrete) probability distribution is ϕ_j and $\phi_j(w_j)$ denotes the probability associated with w_j. Both W_j and ϕ_j are public information. The set of all type profiles is denoted by $W = \Pi_{j \in J} W_j$ and ϕ is the joint probability distribution of $w = (w_1, \ldots, w_n) \in W$, i.e. $\phi(w) = \Pi_j \phi_j(w_j)$. For each job j, let $W_{-j} = \Pi_{k \neq j} W_k$, let $w_{-j} \in W_{-j}$ and let ϕ_{-j} be the corresponding probability distribution. Note that (w_j, w_{-j}) is the type profile where job j has type w_j and the types of all other jobs are w_{-j}.f

In this setting, a mechanism consists of an allocation rule f and a payment scheme π. We consider only direct revelation mechanisms, which are mechanisms in which the only decision made by jobs, is the report of their type. For the remainder of this thesis we denote by w_j a job's true weight and we denote by $\tilde{w}_j$ the reported weight of a job, which may be both true and false. Let w_{-j} and $\tilde{w}_{-j}$ be defined analogously. Based on the reported types, an allocation rule f, taking the role of central authority, chooses a schedule σ. In other words, the allocation rule is a mapping from the set of type profiles to the set of schedules, that is $f : W \to \mathfrak{S}$. Job j is compensated for its waiting time by payment π_j, assigned by the payment scheme π. To express the appreciation of a job for a certain schedule and payment scheme we have to introduce some extra notations. Given job j's waiting time S_j and its weight w_j, it encounters a valuation of $-w_j S_j(\sigma)$ for schedule σ. This means the earlier the better, with a cost of w_j per one unit of time. Additionally receiving a payment π_j, its utility is expressed by $\pi_j - w_j S_j(\sigma)$, i.e. we assume what is called quasi-linear utility [9]. Denote by

$$ES_j(f, \tilde{w}_j) := \sum_{w_{-j} \in W_{-j}} S_j(f(\tilde{w}_j, w_{-j}))\phi_{-j}(w_{-j})$$

the expected waiting time of job j if it reports weight $\tilde{w}_j$ and allocation rule f is applied. Note that $f(\tilde{w}_j, \tilde{w}_{-j}) = \sigma$ and therefore we write $S_j(f(\tilde{w}_j, \tilde{w}_{-j})) =$

[2]Usually the private information of a job is referred to as its type, but since for the 1-dimensional setting the only private information of a job is its weight, for this setting we use weight and type interchangeably.

$S_j(\sigma)$. Let[3]

$$E\pi_j(\tilde{w}_j) := \sum_{w_{-j} \in W_{-j}} \pi_j(\tilde{w}_j, w_{-j})\phi_{-j}(w_{-j})$$

be the expected payment to job j if it reports weight $\tilde{w}_j$.

Definition 1. *A mechanism (f, π) is incentive compatible if jobs have the incentive to report their weight truthfully, i.e. a job obtains highest utility by reporting its true weight. More specifically, a mechanism is:*

- *dominant strategy incentive compatible (DSIC) if for every job j and every two types $w_j, \tilde{w}_j \in W_j$, and any report $\tilde{w}_{-j}$ of other jobs,*

$$\pi_j(w_j, \tilde{w}_{-j}) - w_j S_j(f(w_j, \tilde{w}_{-j})) \geq \pi_j(\tilde{w}_j, \tilde{w}_{-j}) - w_j S_j(f(\tilde{w}_j, \tilde{w}_{-j})). \tag{2.1}$$

  If for allocation rule f there exists a payment scheme π such that (f, π) is DSIC, then f is called implementable in dominant strategies. The payment scheme π is referred to as a dominant strategy incentive compatible payment scheme.

- *Bayes-Nash incentive compatible (BNIC) if for every job j and every two types $w_j, \tilde{w}_j \in W_j$, under the assumption that all jobs apart from j report truthfully,*

$$E\pi_j(w_j) - w_j ES_j(f, w_j) \geq E\pi_j(\tilde{w}_j) - w_j ES_j(f, \tilde{w}_j). \tag{2.2}$$

  If for allocation rule f there exists a payment scheme π such that (f, π) is BNIC, then f is called Bayes-Nash implementable. The payment scheme π is referred to as an Bayes-Nash incentive compatible payment scheme.

Our definition requires jobs to be truthful instead of playing other strategies. This however, is no loss of generality by Myersons' revelation principle [10], as incentive compatible, direct revelation mechanisms can be designed to achieve the same equilibrium payment of any other mechanism[4]. Note

[3]Note that we define $ES_j(f, \tilde{w}_j)$ and $E\pi_j(\tilde{w}_j)$ only for true reports of jobs other than job j. We only need these definitions in a setting where all other jobs report truthfully, as we only consider solutions where truthful reports are an equilibrium. To define $ES_j(f, \tilde{w}_j)$ and $E\pi_j(\tilde{w}_j)$ more generally, would require to take the probability distributions for untruthful reports of $\tilde{w}_{-j}$ of other agents into account.

[4]The proof for the revelation principle for direct revelation mechanisms is as follows. Let us denote by s_j the strategy of job j, i.e. $s_j(w_j)$ is the weight job j reports, given his true weight w_j. Now we can turn any direct revelation mechanism with equilibrium $s = (s_1, \ldots, s_n)$ and allocation rule g in an incentive compatible mechanism, by defining allocation rule $f(t_1, \ldots, t_n) = g(s_1(t_1), \ldots, s_n(t_n))$, i.e. allocation rule $f = g \circ s$ simply simulates the equilibrium strategies of the jobs.

that dominant strategy incentive compatibility is the strongest equilibrium one can ask for. Regardless of w_{-j}, the report of other jobs, reporting its true type is optimal for every job. Bayes-Nash incentive compatibility is a weaker condition and is trivially implied by dominant strategy incentive compatibility. Intuitively, it says the following: given that jobs are risk-neutral and that all jobs apart from job j report truthfully, taking expectations over the possible type profiles of other jobs, it is optimal for job j to report its weight truthfully.

Moreover, rationality of jobs participating in the game is expressed by the following definition.

Definition 2. *A dominant strategy incentive compatible mechanism (f, π) is individually rational (IR) if for every job j, every true type $w_j \in W_j$ and any report $\tilde{w}_{-j}$ of other jobs,*

$$\pi_j(w_j, \tilde{w}_{-j}) - w_j S_j(f(w_j, \tilde{w}_{-j})) \geq 0. \tag{2.3}$$

In other words, individual rationality for DSIC mechanisms implies that the utility of a job reporting its true weight should be positive, regardless what other jobs report. For BNIC mechanisms we have a slightly different definition.

Definition 3. *A Bayes-Nash incentive compatible mechanism (f, π) is individually rational (IRE) if for every job j and every true type $w_j \in W_j$,*

$$E\pi_j(w_j) - w_j E S_j(f, w_j) \geq 0. \tag{2.4}$$

For BNIC mechanisms rationality implies that the expected utility of a job reporting its true weight should be positive. Note that we speak of BNIC mechanisms and therefore the reports of other jobs are assumed to be truthful.

In [5], Heydenreich et al. consider for the scheduling problem so far introduced here, the minimal expected total payment made to the jobs achieved by an allocation rule. For the DSIC setting we assume jobs to maximize their utility, whereas for the DSIC setting we assume jobs to maximize their expected utility.

Definition 4. *An optimal mechanisms (f, π) is a mechanism that is Bayes-Nash incentive compatible, individually rational and minimizes the expected total payment made to jobs. Allocation rule f is called an optimal allocation rule and payment scheme π an optimal payment scheme.*

Heydenreich et al. [5] give an explicit formula for the optimal mechanism in the 1-dimensional setting[5]. The optimal allocation rule f is a modification of Smith's rule. Ergo, for the 1-dimensional scheduling problem the optimal allocation rule is rather simple; scheduling jobs in non-increasing order of weight over processing time ratios, using certain modified weights. Important to mention is that both the modified weights and the payment to a job can be computed using only the characteristics (type and distribution) of the job itself. Furthermore Heydenreich et al. [6] prove that for the 1-dimensional scheduling problem at hand BNIC and DSIC mechanisms in some sense are equivalent. They show there exists a mechanism that is dominant strategy incentive compatible and individually rational, and achieves the same expected total payment as the optimal mechanism defined above.

Manelli and Vincent [8] obtain a similar result for single item auctions with what they refer to as linear utilities. They investigate the model in which a single indivisible object is divided among finitely many agents. The valuation of the agents for the object is private information, although as in our case, from each agent's viewpoint, those valuations are independently distributed according to known distributions. They prove that for this setting, there exists a mechanism that is Bayes-Nash incentive compatible if and only if there exists a dominant strategy incentive compatible mechanism that generates the same expected payments and utilities. For general settings with linear utility and 1-dimensional types Gershkov et al. [4] prove that in settings with only two possible outcomes, e.g. two schedules, BNIC and DSIC is equivalent. However, they also show by counterexample that such an BNIC-DSIC equivalence can only be valid in restrictive environments, as in general, BNIC and DSIC are not equivalent as soon as there are at least three possible outcomes.

2.3 The 2-Dimensional Setting

Having analysed the 1-dimensional setting of the scheduling problem, questions arise whether the results from Heydenreich et al. hold for a setting where both the weight and the processing time of a job are private information. Analogously to the 1-dimensional case, the processing time of job j is some element from the set $P_j = \{p_j^1, \ldots, p_j^{q_j}\}$. For the 2-dimensional setting, t_j, the type of job j is a combination of its weight and processing time and is denoted by (w_j, p_j). The types are drawn from the set $W_j \times P_j$, the type space of job j, also denoted by T_j, according to some publicly known

[5] Optimal mechanism do not need to be unique.

probability distribution ϕ_j. Then $\phi_j(w_j, p_j)$ is the probability associated with type (w_j, p_j). As for the 1-dimensional setting we will denote by t_j and p_j job j's true type and processing time respectively, whereas reports $\tilde{t}_j$ and $\tilde{p}_j$ can be both true and false. The set of all type profiles we denote by $T = \Pi_{j \in J}(W_j \times P_j)$ and $T_{-j} = \Pi_{r \neq j}(W_r \times P_r)$ is the set of type profiles of all jobs except job j. We denote by ϕ the joint probability distribution of $t = (w_1, p_1, \ldots, w_n, p_n) \in T$, ergo $\phi(t) = \Pi_j \phi_j(w_j, p_j)$. Let t_{-j} and ϕ_{-j} be defined analogously. Redefining the type of a job, the expressions for the expected start time and payment to a job also slightly change. Denote by

$$ES_j(f, \tilde{w}_j, \tilde{p}_j) := \sum_{t_{-j} \in T_{-j}} S_j(f((\tilde{w}_j, \tilde{p}_j), t_{-j}))\phi_{-j}(t_{-j})$$

the expected start time of job j when it reports type $(\tilde{w}_j, \tilde{p}_j)$ and allocation rule f is applied. And let[6]

$$E\pi_j(\tilde{w}_j, \tilde{p}_j) := \sum_{t_{-j} \in T_{-j}} \pi_j((\tilde{w}_j, \tilde{p}_j), t_{-j})\phi_{-j}(t_{-j})$$

be the expected payment to job j when it reports type $(\tilde{w}_j, \tilde{p}_j)$. Now the processing time of a job is private information too, jobs have to report both their weight and processing time. Whereas for their weight, jobs could over- and understate it, we make the following assumption on a jobs possible report of its processing time.

Assumption 1. *We assume that jobs can only overstate their processing time, that is, jobs can only report a larger processing time then their true processing time.*

This assumption is made, since reporting a lower processing time than its true processing time can easily be punished by pre-empting the job early (after the reported time). Note that by regarding the processing time of a job as private information, the valuation of a job for a schedule does now also depend on private information of other jobs, namely the processing time of jobs that are scheduled before job j.

For the 2-dimensional setting also the definitions of BNIC and DSIC change.

Definition 5. *A mechanism (f, π) is:*

[6]Note that as for the 1-dimensional setting both $ES_j(f, \tilde{w}_j, \tilde{p}_j)$ and $E\pi_j(\tilde{w}_j, \tilde{p}_j)$ are defined only for truthful reports of other jobs.

- *dominant strategy incentive compatible if for every job j and every two types (w_j, p_j),$(\tilde{w}_j, \tilde{p}_j) \in T_j$ such that $p_j \leq \tilde{p}_j$, and any report $\tilde{t}_{-j}$ of other jobs,*

$$\pi_j((w_j, p_j), \tilde{t}_{-j}) - w_j S_j(f((w_j, p_j), \tilde{t}_{-j})) \geq \pi_j((\tilde{w}_j, \tilde{p}_j), \tilde{t}_{-j}) - w_j S_j(f(\tilde{w}_j, \tilde{p}_j), \tilde{t}_{-j})). \quad (2.5)$$

- *Bayes-Nash incentive compatible if for every job j and every two types (w_j, p_j),$(\tilde{w}_j, \tilde{p}_j) \in T_j$ such that $p_j \leq \tilde{p}_j$, under the assumption that all jobs apart from j report truthfully,*

$$E\pi_j(w_j, p_j) - w_j E S_j(f, w_j, p_j) \geq E\pi_j(\tilde{w}_j, \tilde{p}_j) - w_j E S_j(f, \tilde{w}_j, \tilde{p}_j). \quad (2.6)$$

Note that as for the 1-dimensional setting this definition requires jobs to be truthful instead of playing other strategies. Furthermore also for the 2-dimensional setting, Bayes-Nash incentive compatibility is trivially implied by dominant strategy incentive compatibility. The definition for individual rationality for a DSIC or BNIC mechanism for the 2-dimensional setting read as follows.

Definition 6. *A dominant strategy incentive compatible mechanism (f, π) is individually rational (IR) if for every job j, every true $(w_j, p_j) \in T_j$ and any report $\tilde{t}_{-j}$ of other jobs,*

$$\pi_j((w_j, p_j), \tilde{t}_{-j}) - w_j S_j(f((w_j, p_j), \tilde{t}_{-j})) \geq 0. \quad (2.7)$$

Definition 7. *A Bayes-Nash incentive compatible mechanism (f, π) is individually rational (IRE) if for every job j and every true type $(w_j, p_j) \in T_j$,*

$$E\pi_j(w_j, p_j) - w_j E S_j(f, w_j, p_j) \geq 0. \quad (2.8)$$

As for the 1-dimensional setting we seek to find the minimal expected total payment made to the jobs achieved by an allocation rule. Let f be an allocation rule, then we denote by $E\pi^f(\cdot)$ a payment scheme that minimizes the expected total payment made to the jobs among all payment schemes that make f Bayes-Nash implementable and IRE. Then $E\pi^f(\tilde{t}_j)$ denotes the payment to agent j declaring type $\tilde{t}_j$ under payment scheme $E\pi^f$. Analogously, we denote by $\pi^f(\cdot)$ a payment scheme that minimizes the expected total payment made to the jobs among all payment schemes that make f implementable in dominant strategies and IR. And $\pi_j^f(\tilde{t}_j, \tilde{t}_{-j})$ denotes the

payment to agent j declaring type $\tilde{t}_j$ and given the report of types of other jobs $\tilde{t}_{-j}$, under π^f. Finally $EP^{min}(f)$ and $P^{min}(f)$ denote the corresponding minimal expected total payments made to the jobs, achieved by payment scheme $E\pi^f$ and π^f respectively. Analogously to the 1-dimensional setting, an optimal mechanism is a mechanism that is Bayes-Nash incentive compatible, individually rational and minimizes the expected total payment made $EP^{min}(f)$ to jobs.

As a matter of fact, optimal mechanisms for the 2-dimensional setting do not seem to have a simple characterisation as in the 1-dimensional setting. In order to give a hint towards the intractability of the 2-dimensional optimal mechanism design problem, Heydenreich et al. [5] give an example to show that the optimal mechanism does not in general satisfy a condition known as IIA.

Definition 8. *We say that an allocation rule f satisfies independence of irrelevant alternatives (IIA) if the relative order[7] of any two jobs j_1 and j_2 is the same in the schedules $f(\tilde{t_1})$ and $f(\tilde{t_2})$ for any two type profiles $\tilde{t_1}, \tilde{t_2} \in T$ that differ only in the types of jobs from $J \setminus \{j_1, j_2\}$.*

In other words, an allocation rule f satisfies IIA if the relative order of two jobs is independent of all other jobs. Heydenreich et al. try to show by counterexample that the optimal allocation rule for the 2-dimensional setting does in general not satisfy IIA. In [5] they suggest an instance with three jobs, where the minimal expected payments achieved by an allocation rule that is IIA, exceed the minimal expected payments achieved by an allocation rule that does not satisfy the IIA condition. However, this example was flawed.

The equivalence of BNIC and DSIC for the 2-dimensional setting has not been analysed by Heydenreich et al. and will be discussed in the remainder of this thesis, together with the search for a new example regarding the IIA property of optimal mechanisms.

[7]The relative order of two jobs j_1 and j_2 in a schedule is the position of job j_1 relative to the position of job j_2 and the other way around.

3 Mathematical Programming Formulations

In Section 2 we discussed a single machine scheduling problem that was analysed by Heydenreich et al. The driving questions behind our research have been getting more insight into optimal mechanisms, as well as trying to prove or disprove BNIC-DSIC equivalence for the 2-dimensional setting of this problem. As both questions have been attacked using the same approach, we will first give a detailed description of our approach, where we will later come back to the results.

The initial direction of our research was to find a new instance to prove that the optimal allocation rule for the 2-dimensional setting does in general not satisfy the IIA condition. This would give a hint towards intractability of the optimal mechanism design problem. Creating and checking an instance by hand turned out to be rather difficult and time consuming. Therefore we decided to use a mathematical programming approach, a more systematic approach that has recently become known as automated mechanism design. In the words of Conitzer and Sandholm [2, 13], in automated mechanism design "the mechanism is computationally created for the specific problem instance at hand". An advantage of automated mechanism design over (manual) mechanism design is that it easily can be used in settings beyond those that have been studied using (manual) mechanism design. Furthermore it may yield better mechanisms because the mechanisms are tailored to the specific setting. A disadvantage is that the optimal mechanism design problem has to be solved anew for every instance.

Applying the concept of automated mechanism design we model and solve the optimal mechanism design problem for the 2-dimensional setting by formulating the problem as a mathematical program (MP), to be precise a mixed integer (quadratically constrained) program (MI(QC)P). This allows us to compare different types of mechanisms. Given an instance, the input for the mixed integer program consists of a set of jobs $j \in \{1, \dots, n\}$ with associated types $t_j = (w_j, p_j) \in T_j$ and probabilities $\phi_j(t_j)$ for type t_j, as defined in Section 2. We enumerate type profiles $t = (t_1, \dots, t_n)$ with corresponding probability distribution $\phi(t)$ as well as schedules σ, from which follows σ_i, the position of job i in schedule σ. Having defined these parameters we can calculate $S_{t\sigma j}$, the start or waiting time of job j in schedule σ while the type profile is t. In addition, we introduce binary variables

$$x_{t\sigma} = \begin{cases} 1 & \text{if schedule } \sigma \text{ is assigned to type profile } t \\ 0 & \text{otherwise} \end{cases}$$

for both the BNIC and DSIC setting. In other words, variables $x_{t\sigma}$ pre-

cisely encode the allocation rule of the desired mechanism. Furthermore for the BNIC setting we introduce continuous variables $E\pi_j(t_j)$ and $ES_j(t_j)$ representing the expected payment to and the expected start time of job j, reporting type t_j respectively. For the DSIC setting we only introduce continuous variables $\pi_{t\sigma j}$ representing the payment to job j given the overall type profile t and schedule σ. A solution of the integer program is an allocation rule, together with a corresponding payment scheme.

3.1 Bayes-Nash Implementations

The problem of finding an optimal mechanism, i.e. a mechanism that is Bayes-Nash incentive compatible, individually rational and among such mechanisms minimizes the expected total payments that have to be made to the jobs, can be represented by the following mixed integer program.

$$\min \sum_j \sum_{t_j} \phi_j(t_j) E\pi_j(t_j) \tag{3.1a}$$

$$\sum_\sigma x_{t\sigma} = 1 \qquad \forall\, t \tag{3.1b}$$

$$ES_j(t_j) = \sum_{t_{-j},\sigma} S_{t\sigma j} x_{t\sigma} \phi_{-j}(t_{-j}) \qquad \forall\, j, t_j \tag{3.1c}$$

$$E\pi_j(t_j) \geq w_j(t_j) ES_j(t_j) \qquad \forall\, j, t_j \tag{3.1d}$$

$$\begin{aligned} E\pi_j(t_j^i) \geq\ & E\pi_j(t_j^k) \\ & - w_j(t_j^i) \left(ES_j(t_j^k) - ES_j(t_j^i) \right) \end{aligned} \qquad \forall\, j, (t_j^i, t_j^k) \in T^{bn} \tag{3.1e}$$

$$x_{t\sigma} \in \{0,1\} \tag{3.1f}$$

$$E\pi_j, ES_j \geq 0 \tag{3.1g}$$

Continuous variables $E\pi_j(t_j)$ denote the expected payment to job j having type t_j, as defined in Section 2.3. Together they define the payment scheme of the Bayes-Nash incentive compatible optimal mechanism. Objective (3.1a) minimizes the expected total payments made to all jobs, whereas constraints (3.1b) enforce a feasible allocation rule by assigning exactly one schedule to each type profile. Note that the notation of $ES_j(t_j)$, the variable for the expected start time of a job, is slightly different from the notation in Section 2.3. In Section 2.3 the expected start time was explicitly defined only for allocation rule f and type t_j, where the same result is now obtained in (3.1c) by summing also over all schedules σ and multiplying by the corresponding binary $x_{t\sigma}$ variable.

The individual rationality and incentive constraints are represented by (3.1d) and (3.1e) respectively, although the latter are somewhat rearranged[8]. According to Definition 5 the incentive constraints only hold for pairs of (t_j^i, t_j^k) for which processing time of job j having type t_j^i, is smaller or equal to processing time of job j having type t_j^k, as jobs cannot overstate their processing time. This is implemented by defining constraints (3.1e) only for $(t_j^i, t_j^k) \in T^{bn}$ where

$$T^{bn} = \{(t_j^i, t_j^k) \mid p_j(t_j^i) \leq p_j(t_j^k)\}.$$

Finally constraints (3.1f) express the integrality of the $x_{t\sigma}$ variables and constraints (3.1g) express the bounds on the expected payment to and expected start time of job j.

3.2 Dominant Strategy Implementations

To compare BNIC and DSIC mechanisms, we build a mathematical program for the DSIC setting, too. The problem of finding a mechanism that is dominant strategy incentive compatible, individually rational and among such mechanisms minimizes the expected total payments that have to be made to the jobs, can be represented by the following mixed integer program.

$$\min \sum_t \sum_\sigma \sum_j \phi(t)\pi_{t\sigma j} \tag{3.2a}$$

$$\sum_\sigma x_{t\sigma} = 1 \qquad \forall\, t \tag{3.2b}$$

$$\pi_{t\sigma j} \geq w_j(t) S_{t\sigma j} x_{t\sigma} \qquad \forall\, t, \sigma, j \tag{3.2c}$$

$$\pi_{t\sigma j} \geq \pi_{t'\sigma' j} - w_j(t)(S_{t'\sigma' j} - S_{t\sigma j}) + M(x_{t\sigma} - 1) \qquad \forall\, \sigma, \sigma', j, (t, t') \in T^{ds} \tag{3.2d}$$

$$x_{t\sigma} \in \{0, 1\} \tag{3.2e}$$

$$\pi_{t\sigma j} \geq 0 \tag{3.2f}$$

Note that this MIP is very similar to the one discussed in Section 3.1. There are however some important dissimilarities concerning the variables and the individual rationality and incentive constraints. In (3.2), the variables for the expected payments are replaced by variables $\pi_{t\sigma j}$, representing the payment to job j given the overall type profile t and schedule σ. As we still

[8] Note that we introduce $w_j(t_j)$, representing the weight of job j when having type t_j.

seek to minimize the expected total payments made to the jobs, the objective is represented by (3.2a). Intuitively one would say that we only need to define payments for all type profiles t and jobs j, as ultimately to each type profile will be assigned only one schedule σ. However, up front we do not know which schedule is assigned to which type profile and therefore we also have to introduce payments for all schedules σ. The constraints that enforce a feasible allocation rule remain unchanged and are represented by (3.2b). In the DSIC setting we no longer need expected start times and therefore constraints (3.1c) can be omitted and the individual rationality constraints translate to (3.2c). These constraints[9] are stronger than constraints (3.1d), as individual rationality has to hold not only in expectation, but for every type profile - schedule combination that is chosen by the allocation rule.

In addition to changing the start time and payment variables, we have to introduce a big-M construction for the rearranged incentive constraints (3.2d). This is to enforce that the incentive constraints are tight for type profile - schedule combinations that are chosen by the allocation rule and are trivially fulfilled otherwise[10]. As for the BNIC setting jobs cannot overstate their processing time, constraints (3.2d) only hold for pairs of type profiles (t, t') such that the processing time of job j under type profile t is smaller or equal than its processing time under type profile t'. Also the reported types of other jobs must be equal in both t and t'. Even more, when the type profiles are identical, the incentive constraints in combination with the MP are fulfilled trivially, hence we do not define them when $t = t'$. A schematic representation of T^{ds}, the set for which the incentive constraints are defined, can be found in Table 1. In this table $+$, $-$ and $+-$ represent that the corresponding condition is fulfilled, not fulfilled or either of both, respectively. Note that for type profile-schedule combinations not chosen by the allocation rule, both constraints (3.2c) and (3.2d) are trivially fulfilled. Therefore, for these combinations, the payments are automatically set to 0 due to the minimizing objective and for each t we are left with only one σ for which $\pi_{t\sigma j} > 0$. Finally constraints (3.2e) express the integrality of the $x_{t\sigma}$ variables and constraints (3.2f) express the bounds on the payment to job j when the type profile is t and schedule σ is chosen.

[9]Note that we introduce $w_j(t)$, representing the weight of job j when the overall type profile is t.

[10]Note that in the experiments we did not tune M to a small value in order to tighten the constraints. Instead we have set it to a value of 1000 to make it sufficiently large.

$t_j \neq t'_j$	$t_{-j} = t'_{-j}$	$p_j(t) \leq p_j(t')$	$\in \boldsymbol{T^{ds}}$
$-$	$+$	$+$	$-$
$+$	$+$	$+$	$+$
$+-$	$-$	$+-$	$-$
$+-$	$+-$	$-$	$-$

Table 1: Schematic representation of T^{ds}, the set of type profile pairs for which the dominant strategy incentive constraints are defined.

3.3 Independence of Irrelevant Alternatives

Using mathematical programs (3.1) and (3.2) we can compare BNIC and DSIC mechanisms. However, to check whether the optimal allocation rule does in general satisfy IIA, we must be able to add constraints to MIP (3.1) that imply the IIA condition[11]. By adding the following quadratic constraints

$$x_{t\sigma}x_{t'\sigma'} \leq (\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i)x_{t\sigma}x_{t'\sigma'} \qquad \forall\, (t,t') \in T^{iia}, \sigma, \sigma', j, i \neq j \quad (3.3)$$

to the mathematical formulation of the Bayes-Nash optimal mechanism design problem, the solution of the modified program is an allocation rule that satisfies IIA, together with a corresponding payment scheme[12].

The constraints are based on the fact that if the relative order of job j and job i in schedule σ and σ' is different, i.e. $(\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i) < 0$, then not both $x_{t\sigma}$ and $x_{t'\sigma'}$ can equal 1. However, we must pay attention for which pairs (t, t') we define the IIA constraints. By definition of IIA we are interested only in pairs (t, t') that differ only in the types of jobs from $J \setminus \{i, j\}$, so $t_j = t'_j$ and $t_i = t'_i$. Furthermore, if also $t = t'$, constraints (3.3) in combination with the MIP are fulfilled trivially and therefore we only define the IIA constraints for $t \neq t'$. The set of pairs (t, t') that have these properties we denote by T^{iia}. A schematic representation T^{iia} can be found in Table 2. But even more, if $i = j$ we would set either $x_{t\sigma} = 0$ or $x_{t'\sigma'} = 0$, although the relative order of the job j and job i is identical and

[11] Not to by adding the IIA constraints to MIP (3.2), one might also try to (dis)prove that the allocation rule that is implementable in dominant strategies and minimizes the total expected payments made to jobs, satisfies the IIA condition. However, that is behind the scope of this thesis.

[12] After the research was finished, we found out that the same result can be retrieved by adding constraints $x_{t\sigma} + x_{t'\sigma'} \leq 1$ for the same $t, t', \sigma, \sigma', j, i$. This substitution leads to a linear program also for the IIA condition and therefore to shorter running times. However these results have not been added to this report.

we would make the MP infeasible. Therefore the IIA constraints are defined only when $(t, t') \in T^{iia}$ and $i \neq j$. Finally note that by adding the IIA constraints to MIP (3.1), the MIP changes to a mixed integer quadratically constraint program (MIQCP).

$t_j = t'_j$	$t_i = t'_i$	$t \neq t'$	$\in \boldsymbol{T^{iia}}$
$+$	$+$	$+$	$+$
$-$	$+-$	$+-$	$-$
$+-$	$-$	$+-$	$-$
$+-$	$+-$	$-$	$-$

Table 2: Schematic representation of T^{iia}, the set of type profile pairs for which the IIA constraints are defined.

3.4 Implementation of MP Formulations

The MIPs and MIQCPs proposed in the previous section are solved using ILOG CPLEX. This is a tool for solving several kinds of mathematical optimization problems, among them MIPs and MIQCPs. It is folklore that reformulating constraints or even the complete mathematical program can have a substantial influence on the solving time of the MP, therefore we seek to 'massage' the MPs in order to minimize the solving time. One can think of adding or removing constraints and eliminating variables. At default settings, ILOG CPLEX uses several techniques to preprocess problems by simplifying constraints, reducing problem size, and eliminating redundancy and symmetry. However, these techniques can be seen as a black-box and therefore we additionally tried to modify the MPs ourself to improve the solving time.

One of the most obvious modifications is to get rid of quadratic constraints (3.3). There are several ways to linearise these constraints and we use a very basic one. We replace all products $x_{t\sigma}x_{t'\sigma'}$ in the program by binary variables $y_{t\sigma t'\sigma'}$, so that we obtain

$$y_{t\sigma t'\sigma'} \leq (\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i)y_{t\sigma t'\sigma'} \qquad \forall\, (t, t') \in T^{iia}, \sigma, \sigma', j, i \neq j \tag{3.4}$$

as new IIA constraints. To assure that the variables $y_{t\sigma t'\sigma'}$ attain the same value as the original product $x_{t\sigma}x_{t'\sigma'}$ we have to add the following con-

straints.

$$y_{t\sigma t'\sigma'} \leq x_{t\sigma} \qquad \forall\, t,\sigma,t',\sigma' \qquad (3.5a)$$

$$y_{t\sigma t'\sigma'} \leq x_{t'\sigma'} \qquad \forall\, t,\sigma,t',\sigma' \qquad (3.5b)$$

$$y_{t\sigma t'\sigma'} \geq x_{t\sigma} + x_{t'\sigma'} - 1 \qquad \forall\, t,\sigma,t',\sigma' \qquad (3.5c)$$

We can define the $y_{t\sigma t'\sigma'}$ variables for all t,σ,t' and σ' as in (3.5). However, we only need them to replace all $x_{t\sigma}x_{t'\sigma'}$ products occurring in the MP. Therefore we define the y variables and corresponding constraints (3.5) only for t and t' for which $(t,t') \in T^{iia}$.

In constraints (3.3) there is a lot of symmetry since $x_{t\sigma}x_{t'\sigma'} = x_{t'\sigma'}x_{t\sigma}$. Replacing these quadratic constraints by introducing y variables, the symmetry remains ($y_{t\sigma t'\sigma'} = y_{t'\sigma' t\sigma}$). To get rid of this symmetry, we define both the quadratic and linearised IIA constraints as well as the y variables and constraints (3.5) only for t,σ,t',σ' such that $t \cdot |\mathfrak{S}| + \sigma < t' \cdot |\mathfrak{S}| + \sigma'$, assuming that the types and schedules are ordered in some arbitrary way. This way, we reduce the number of variables and constraints by roughly a factor two. Note that we do not define them when $t \cdot |\mathfrak{S}| + \sigma = t' \cdot |\mathfrak{S}| + \sigma'$, as this implies that $t = t'$, hence $(t,t') \notin T^{iia}$.

With our last modification we can even further reduce the number of IIA constraints. Let us consider the value of $(\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i)$ for different σ,σ',i,j. At this point for all cases where $(\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i) \neq 0$ the IIA constraints are defined. However, $(\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i) > 0$ implies $(\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i) \geq 1$ as σ_i is integer and in this case the IIA constraints are redundant. Therefore we need to define constraints (3.3) and (3.4) only for $(\sigma_j - \sigma_i)(\sigma'_j - \sigma'_i) < 0$.

4 Solution Method

In the previous section we formulated mathematical programs for different types of mechanisms in the 2-dimensional scheduling problem. In this section we give an extensive description of the method we use to compare BNIC and DSIC mechanism, fulfilling the IIA constraints or not, whereas the computational results themselves can be found in Section 5.

First we systematically generate instances at random, which we use for our research. Then we construct, using the instances, the proper mathematical programs and finally we optimally solve the MPs to find answers to our research questions mentioned in Section 3. All components have been realized in a program written in C++, which successively executes the components, one by one described in this section.

4.1 Generating Instances

Our first step is to generate instances, which are necessary to test our hypotheses on. The size of the instances we generate depends on the research question we focus on. As the definition of IIA is irrelevant for instances with one or two jobs, for this part of the research we only generate instances with three or more jobs. To investigate similarities between BNIC and DSIC we also generate instances with two jobs. We do not want the solving time of the MPs to be too long and therefore we initially focus on smaller instances. If we do not find any results using these instances we subsequently increase the number of jobs and/or types of a job.

The input arguments of the C++ program consist of the number of jobs and if desired the number of types per job. One can also decide to generate instances with a random number of jobs or types per job. Given the number of jobs and types per job, the program generates for each type of a job a corresponding weight, processing time and probability.

The weights and processing times of a job chosen first, are chosen uniformly over the discrete integer set $\{1, \ldots, 10\}$. However, to ensure that we do not encounter equal weights or processing times for a job, we iteratively generate a weight or processing time and delete its value from the set from which we chose the last value. From the new set we obtain, we again select uniformly a new weight or processing time. We will denote this type of distribution by (discrete, 'uniform'). The probabilities of the types of a job cannot be chosen randomly over a discrete set, as the sum over the probabilities of the types of a job must be equal to 1. Instead, per job j, for each jobtype t_j we select uniformly a number v_j^i from the discrete integer

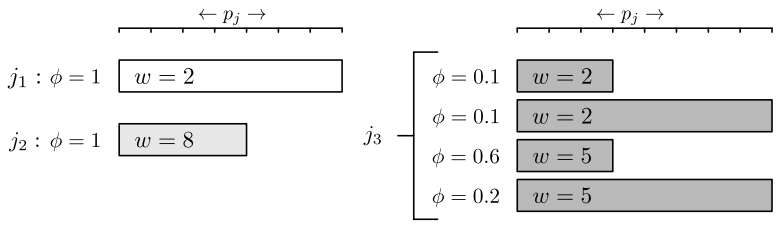

Figure 1: Graphical representation of instance in Appendix B.

set $\{0, \ldots, 100\}$. Afterwards we divide the selected numbers v_j^i by $\sum_i v_j^i$, the sum of all numbers chosen for the jobtypes of job j. This leaves us with probabilities $\phi_j(t_j)$, which all have the same expected value. However, eventually we only need instances with probabilities rounded to one-tenths or one-hundredths. Therefore we round down all probabilities except for the last one, which we set to the remaining probability[13]. We will denote the obtained distribution by (discrete, $E(t_j) \approx E(t'_j)$). Note that initially we use probabilities rounded to one-tenths, whereas we round the probabilities to one-hundredths if we do not obtain results with the former. An overview of the type, range and distribution of the elements of a job's type can be found in Table 3.

Property	Symbol	Type	Range	Distribution
Weight	w_j	Integer	$[1, 10]$	Discrete 'uniform'
Processing time	p_j	Integer	$[1, 10]$	Discrete 'uniform'
Probability type	$\phi_j(t_j)$	Decimal	$[0.0, 1.0]$	Discrete, $E(t_j) \approx E(t'_j)$
Probability type	$\phi_j(t_j)$	Hundredths	$[0.00, 1.00]$	Discrete, $E(t_j) \approx E(t'_j)$

Table 3: Type, range and distribution of the elements of a type of a job.

To select 'random' numbers from some discrete set, we use the *rand*() function in C++. This is a pseudo-random integral number generator containing an algorithm that returns a sequence of apparently non-related numbers each time it is called. The algorithm uses a seed to generate the series, which should be initialized to some distinctive value using *srand*(). Choosing the same seed value will lead to the same 'random' sequence and thus makes it possible to reproduce the experiments any time. An example of an instance we generate for our research is sketched in Figure 1.

Once proven or disproven that optimal mechanisms do in general satisfy

[13]Note that in this way we 'pollute' the expected values E.

the IIA condition, or that BNIC and DSIC in general is equivalent, we seek to specify our general results. To test our results on more specific instances, we therefore generate instances with specific properties. One type of instances we investigate are instances for which the types of job have a product distribution.

Definition 9. *A product distribution is a probability distribution of which the marginal distributions are pairwise stochastically independent.*

In the 2-dimensional scheduling problem the pair (w_j, p_j) equals t_j, the type of job j. Suppose that the weight and processing time of job j have probability distribution φ_j and ψ_j respectively[14]. Then ϕ_j, the probability distribution associated with type t_j, is a product distribution if for all $w_j \in W_j$ and $p_j \in P_j$ holds $\phi_j(w_j, p_j) = \varphi_j(w_j) \cdot \psi_j(p_j)$. In other words, ϕ_j is a product distribution if the probability associated with having type (w_j, p_j) is the product of the marginal distributions φ and ψ for having weight w_j and processing time p_j, respectively.

4.2 Instance File Format

The information of an instance is written to a text file in a specific format. The first line contains only the number of jobs, whereas the second line contains the number of types per job, separated by a white space. The remainder of the text file consists of a single line for each type, the types sorted per job, in ascending order of weight and processing time. These lines state the weight, processing time and probability of a type of a job respectively, each separated by a white space. In Appendix B can be found the text file corresponding to the instance shown in Figure 1.

4.3 Computational Procedure and Details

After generating an instance and constructing its corresponding text file, the C++ program builds from this text file a mathematical program for two types of mechanisms we would like to compare. The MPs are solved using ILOG CPLEX v12.2.0.0 on a computer equipped with an Intel Core Duo processor P9500 at 2.53GHz and 4GB RAM under WINDOWS operating system. To be able to solve the MPs, the C++ program constructs a CPLEX LP format file, which is a file format in which one can enter a problem in

[14]Note that distributions φ_j and ψ_j are generated in the same way as distribution $\phi_j(t_j)$ in Table 3.

a natural, algebraic LP formulation (see Appendix C[15]). As we want to compare different types of mechanisms, e.g. BNIC or DSIC, fulfilling the IIA constraints or not, one can give as input arguments the settings to be compared.

Surprisingly, it turned out that the solving time of the mathematical program with quadratic IIA constraints (3.3) is in many cases smaller than the solving time of the mathematical program with linearised IIA constraints. For smaller instances the solving time is of the same order, whereas for larger instances with more jobs or jobtypes, the solving time of the MIP can become on average more than 10 times larger that of the MIQCP. Even more, the size of the LP files for the MIP is much larger than that of the MIQCP. Therefore, both for the BNIC and DSIC setting, we use the quadratic IIA constraints. The remaining modifications to the MP mentioned in Section 3.4 are all applied as they all contribute to a small improvement in the solving time.

The two LP files containing the MPs for the types of mechanisms we would like to compare, we solve calling CPLEX from within the C++ program using the ILOG CPLEX callable library. Whenever the program runs into an instance for which there is a difference in objective between the two MPs, the program saves both the instance and the corresponding LP files. Finally, after finding a specified number of such instances, the program terminates and we can analyse the results we found. The Pseudo-code of the complete C++ program is sketched in Appendix D.

In Table 4 we show some computational properties for different instances and different types of mechanisms. The properties we consider here are:

- In the first column we report $|J|$, the number of jobs of the instance we consider. Note that the number of different schedules for this instance is $|J|!$.
- Then we report the number of jobtypes t_j for each job j. Here we denote by $1-1-4$ an instance with three jobs, where both job 1 and 2 have one type and job 3 has four types. This means that in total there are $1 \cdot 1 \cdot 4 = 4$ different type profiles.
- In column three we report the setting we consider, where 'BN' denotes the Bayes-Nash incentive compatible setting and 'DS' denotes the dominant strategy incentive compatible setting.
- Whether we consider the IIA constraints and if so, which IIA constraints we add to the MP, can be found in column four. By 'no'

[15]This CPLEX file corresponds to the BNIC setting of the instance shown in Figure 1 and in Appendix B.

# Jobs	# Jobtypes	Setting	IIA	# Var	# Constr	LP (kb)	Build (ms)	Read (ms)	Solve (ms)
2	$1-4$	BN	no	18	22	1.1	1.1	1.5	2.6
2	$1-4$	DS	no	24	52	2.7	1.2	1.6	5.1
2	$4-4$	BN	no	48	48	3.0	1.2	1.6	3.5
2	$4-4$	DS	no	96	336	17.5	2.2	2.1	22.4
2	$6-6$	BN	no	96	102	6.8	2.4	2.3	8.1
2	$6-6$	DS	no	216	1188	64.0	7.6	5.4	177.7
3	$1-1-4$	BN	no	36	24	2.22	1.3	1.6	2.8
3	$1-1-4$	BN	quad	36	24	5.49	1.6	2.1	15.9
3	$1-1-4$	BN	lin	276	588	21.26	2.7	2.2	15.5
3	$1-1-4$	DS	no	96	364	18.78	2.0	1.9	11.5
3	$1-1-6$	BN	no	52	43	3.66	1.1	1.6	3.1
3	$1-1-6$	BN	quad	52	43	11.83	2.3	3.3	39.8
3	$1-1-6$	BN	lin	628	1429	51.88	5.7	3.4	30.4
3	$1-1-6$	DS	no	144	870	44.47	3.8	3.2	31.8
3	$1-4-4$	BN	no	114	50	7.5	2.1	2.5	7.5
3	$1-4-4$	BN	quad	114	50	34.3	6.3	6.3	214.8
3	$1-4-4$	BN	lin	4530	12242	468.7	53.1	29.1	529.4
3	$1-4-4$	DS	no	384	2608	138.9	13.6	9.1	1343.4
3	$1-6-6$	BN	no	242	104	17.1	3.3	3.1	11.9
3	$1-6-6$	BN	quad	242	104	122.9	15.5	15.8	963.3
3	$1-6-6$	BN	lin	23138	65120	2558.7	213.3	122.0	6287.5
3	$4-4-4$	BN	no	408	112	28.7	4.3	4.5	13.1
3	$4-4-4$	BN	quad	408	112	199.2	28.1	26.5	1799.0
3	$4-4-4$	BN	lin	73368	213040	8684.4	927.9	588.0	72031.5

Table 4: Computational properties for different instances and different types of mechanisms.

we mean that we do not consider IIA constraints, whereas linear and quadratic IIA constraints are represented by 'lin' and 'quad' respectively.

- In column five and six we state the number of variables and constraints for the MP of the corresponding setting. Note that in the number of constraints we do not count the objective and the bounds on the variables.
- The size of the ILOG CPLEX LP file which is build by our C++ program can be found in column seven. Note that the size of the LP is in kilobyte.
- In the last three columns we state the average build, read and solve time of an instance in seconds. By build time we mean the time it takes for C++ to build the corresponding CPLEX LP file for a given instance. The time it takes for CPLEX to read the LP file we denote by the read time. Finally the solve time is the time it takes CPLEX to solve the MP that is in the LP file. Note that for instances with 4 or more jobs and more than 16 type profiles, except for the MP for the Bayes-Nash setting without IIA constraints, CPLEX runs out of memory. For smaller settings the average is taken over 1000 instances, whereas for settings larger than the setting with 3 jobs, and $1-4-4$ jobtypes, we take the average over 100 instances.

5 Computational Results

With use of the mathematical programs proposed in Section 3, which we have implemented as discussed in Section 4, we are able to test two main hypotheses for the 2-dimensional scheduling problem. This resulted in a proof by counterexample that the optimal allocation rule in the 2-dimensional setting in general does not satisfy the IIA condition, as well as in a counterexample which proofs that BNIC and DSIC for the 2-dimensional setting in general is not equivalent. In this section we address both main results as well as some side results.

5.1 Optimal Mechanisms and IIA

First we discuss the result on the optimal allocation rule for the 2-dimensional scheduling problem, concerning the IIA condition. By formulating the optimal mechanism design problem as MP, and generating problem instances at random, we have been lucky in finding an instance for which for the optimal allocation rule, the relative order of two jobs is dependent of the other jobs. Thus, we have ultimately found a proof for the following theorem, stated in [5].

Theorem 1. *The optimal allocation rule for the 2-dimensional setting does in general not satisfy independence of irrelevant alternatives.*

Proof. Consider the following instance with three jobs. Both job 1 and job 2 have a type space containing only one type, type $(w_1, p_1) = (2, 1)$ and $(w_2, p_2) = (9, 8)$ respectively. Job 3 has type space $(w_3, p_3) \in \{1, 3, 5\} \times \{5, 7\}$ and the corresponding probabilities for its types are listed below.

$$\begin{array}{lll} \phi_3(1,5) = 0.24 & \phi_3(1,7) = 0.24 & \phi_3(3,5) = 0.02 \\ \phi_3(3,7) = 0.24 & \phi_3(5,5) = 0.16 & \phi_3(5,7) = 0.10 \end{array}$$

A graphical representation of the instance is sketched in Figure 2. We will show that for this instance the unique allocation rule that is Bayes-Nash implementable, individually rational and minimizes the expected total payments, does not satisfy independence of irrelevant alternatives. The instance has three jobs and therefore we have $3! = 6$ different schedules. We denote by schedule 213 the schedule where job 2 is scheduled first and job 3 is scheduled last. Job 1 and job 2 both have only one possible type, whereas job 3 can have 6 types. Therefore the type profile is only dependent on the

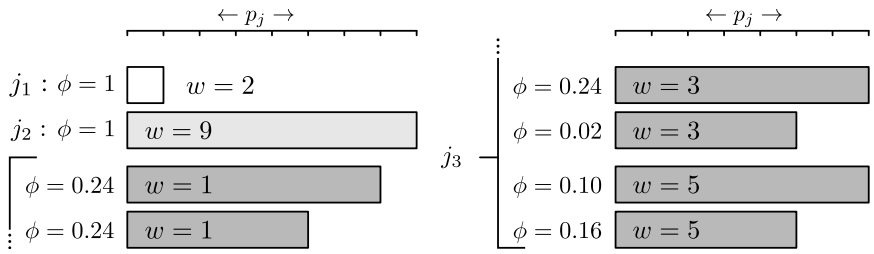

Figure 2: Graphical representation of the instance for which the optimal allocation rule does not satisfy the IIA condition.

type of job 3. We will denote these six type profiles by different cases

$$\begin{aligned} &\text{case } a : (w_3, p_3) = (1, 5) && \text{case } b : (w_3, p_3) = (1, 7) \\ &\text{case } c : (w_3, p_3) = (3, 5) && \text{case } d : (w_3, p_3) = (3, 7) \\ &\text{case } e : (w_3, p_3) = (5, 5) && \text{case } f : (w_3, p_3) = (5, 7). \end{aligned}$$

For this instance IIA implies that for all six cases, i.e. type profiles, the allocation rule must choose a schedule in which the relative order of job 1 and job 2 is the same[16]. Therefore the allocation rule must choose schedules from either $\{123, 132, 312\}$ or $\{213, 231, 321\}$ for all six cases.

As an example we compute the minimal expected total payments achieved by allocation rule f, that assigns the following schedules to reported types.

$$\begin{aligned} &\text{case } a \to 123 && \text{case } b \to 123 && \text{case } c \to 132 \\ &\text{case } d \to 123 && \text{case } e \to 132 && \text{case } f \to 312 \end{aligned}$$

Since we consider a BNIC setting, we take into account individual rationality constraints (2.8) and incentive constraints (2.6). For job 1 and 2 we do not need to take into account incentive constraints as they only have one type. In order to evaluate the individual rationality constraints we need to compute the expected start times for job 1 and 2. These are computed by considering the start time of the job in the schedules assigned to the six cases and account for the probability for each case. For job 1 the minimal expected payment that is enforced by the individual rationality constraint, equals

$$\begin{aligned} E\pi_1^f(2,1) &= w_1 \cdot ES_1(f, 2, 1) \\ &= 2 \cdot (0.24 \cdot 0 + 0.24 \cdot 0 + 0.02 \cdot 0 + 0.24 \cdot 0 + 0.16 \cdot 0 + 0.10 \cdot 7) \\ &= 1.40 \end{aligned}$$

[16] We do not need to consider the relative order of job 3 as job 1 and job 2 only have one type.

whereas for job 2 we have

$$\begin{aligned} E\pi_2^f(9,8) &= w_2 \cdot ES_2(f,9,8) \\ &= 9 \cdot (0.24 \cdot 1 + 0.24 \cdot 1 + 0.02 \cdot 6 + 0.24 \cdot 1 + 0.16 \cdot 6 + 0.10 \cdot 8) \\ &= 23.40. \end{aligned}$$

For job 3 we have to take into account both the incentive and the individual rationality constrains, both implying a lower bound on the expected payments. Note that although we still refer to them as expected start times, the start times for job 3 are not expected since job 1 and job 2 have only 1 type. Therefore we only have to consider the start time of job 3 in the schedule assigned to the corresponding case. Individual rationality (IRE) for job 3 requires

$$\begin{aligned} &E\pi_3^f(1,5), E\pi_3^f(1,7) \geq 1 \cdot 9 = 9 \\ &E\pi_3^f(3,7) \geq 3 \cdot 9 = 27 \\ &E\pi_3^f(3,5) \geq 3 \cdot 1 = 3 \\ &E\pi_3^f(5,5) \geq 5 \cdot 1 = 5 \\ &E\pi_3^f(5,7) \geq 5 \cdot 0 = 0. \end{aligned}$$

There are 21 incentive constraints[17] and therefore it may seem hard to check if all constraints are fulfilled. However, due to the individual rationality constraints (2.8), the left-hand side of incentive constraints (2.6) is trivially greater or equal to zero. Therefore we only need to consider incentive constraints for which the right-hand side is greater than zero. For these constraints we modify the payments in order to fulfil the incentive constraints. This will influence also the right-hand side of the other incentive constraints and therefore we iteratively check all incentive constraints for which the right-hand side is greater than zero and modify the payments until all incentive constraints are fulfilled[18]. In our first iteration we observe that the

[17] Remember that we assume that jobs can only overstate their processing time.

[18] In [5] Heydenreich et al. compute the minimal (expected) payments for an allocation rule f by searching for the shortest path in a type graph T^f of a job. In this sense our 'algorithm' is the Bellman-Ford algorithm [3] where we iteratively try to relax all edges.

only incentive constraints that remain are

$$\begin{aligned}
E\pi_3^f(1,5) - 1 \cdot ES_3(f,1,5) &\geq E\pi_3^f(3,5) - 1 \cdot ES_3(f,3,5) \\
E\pi_3^f(1,5) - 1 \cdot ES_3(f,1,5) &\geq E\pi_3^f(3,7) - 1 \cdot ES_3(f,3,7) \\
E\pi_3^f(1,5) - 1 \cdot ES_3(f,1,5) &\geq E\pi_3^f(5,5) - 1 \cdot ES_3(f,5,5) \\
E\pi_3^f(3,5) - 3 \cdot ES_3(f,3,5) &\geq E\pi_3^f(5,5) - 3 \cdot ES_3(f,5,5) \\
E\pi_3^f(1,7) - 1 \cdot ES_3(f,1,7) &\geq E\pi_3^f(3,7) - 1 \cdot ES_3(f,3,7).
\end{aligned}$$

By setting $E\pi_3^f(1,5), E\pi_3^f(1,7) \geq 27$ and $E\pi_3^f(3,5) \geq 5$ these constraints are fulfilled and in our second iteration we notice that no right-hand side of any incentive constraint is greater than zero. Therefore the minimal payments to job 3 are

$$\begin{aligned}
E\pi_3^f(1,5) &= E\pi_3^f(1,7) = E\pi_3^f(3,7) = 27 \\
E\pi_3^f(3,5) &= E\pi_3^f(5,5) = 5 \\
E\pi_3^f(5,7) &= 0.
\end{aligned}$$

Now we have computed the minimal expected payments to all jobs, we can compute the minimal expected total payment achieved by allocation rule f.

$$\begin{aligned}
EP^{min}(f) &= 1 \cdot E\pi_1^f(2,1) + 1 \cdot E\pi_2^f(9,8) + \sum_{t_3 \in T_3} \phi_3(t_3) E\pi_3^f(t_3) \\
&= 1.40 + 23.40 + 0.24 \cdot 27 + 0.24 \cdot 27 + 0.02 \cdot 5 \\
&\quad + 0.24 \cdot 27 + 0.16 \cdot 5 + 0.10 \cdot 0 \\
&= 45.14
\end{aligned}$$

In the same way we can compute the minimal expected total payments achieved by all other $2 \cdot 3^6 - 1 = 1457$ allocation rules that are IIA. For this instance it turns out that allocation rule f is the unique Bayes-Nash implementable allocation rule that achieves minimal expected total payments, while satisfying the IIA condition.

Now consider allocation rule g, that chooses for each case/type profile the following schedule.

case $a \to 123$	case $b \to 123$	case $c \to 231$
case $d \to 123$	case $e \to 132$	case $f \to 312$

This allocation rule clearly does not satisfy the IIA condition as the relative order of job 1 and job 2 in the schedule chosen for case c is different from

the relative order in the schedule chosen for the rest of the cases, although the types of job 1 and job 2 are for all cases identical. Using exact the same approach as for allocation rule f, we can calculate the payment scheme corresponding to allocation rule g. For job 1 and 2 again only considering the individual rationality constraints, the minimal payments to job 1 and job 2 are $E\pi_1^g(2,1) = 1.92$ and $E\pi_2^g(9,8) = 22.32$. Computing by the individual rationality constraints for job 3 a lower bound on the expected payments and again iteratively checking the incentive constraints, leads to

$$\begin{array}{lll} E\pi_3^g(1,5) \geq 27 & E\pi_3^g(1,7) \geq 27 & E\pi_3^g(3,5) \geq 26 \\ E\pi_3^g(3,7) \geq 27 & E\pi_3^g(5,5) \geq 5 & E\pi_3^g(5,7) \geq 0. \end{array}$$

For allocation rule g the minimal expected total payment is

$$\begin{aligned} EP^{min}(g) &= 1 \cdot E\pi_1^g(2,1) + 1 \cdot E\pi_2^g(9,8) + \sum_{t_3 \in T_3} \phi_3(t_3) E\pi_3^g(t_3) \\ &= 1.92 + 22.32 + 0.24 \cdot 27 + 0.24 \cdot 27 + 0.02 \cdot 26 \\ &\quad + 0.24 \cdot 27 + 0.16 \cdot 5 + 0.10 \cdot 0 \\ &= 45.00. \end{aligned}$$

This proves the claim. □

Now Theorem 1 has been proven, an obvious question is for which specific instances the optimal allocation rule for the 2-dimensional scheduling problem does satisfy the IIA condition. A type of instance we have investigated, is one in which jobtypes have a product distribution (see Definition 9). For this type of instance we were not able to find an example for which the optimal allocation rule does not satisfy the IIA condition.

Conjecture 1. *For the 2-dimensional scheduling problem, where the types of a job have a product distribution, the optimal allocation rule satisfies the IIA condition.*

Despite extensive computational research, we were not able to find an instance contradicting Conjecture 1. Further research will have to verify our conjecture.

5.2 BNIC-DSIC Equivalence

Our second research question is whether for the 2-dimensional single machine scheduling problem, there always is a DSIC mechanism that achieves the same expected total payment as the optimal BNIC mechanism. Using exact

the same solution method as in the previous section, we came to the following result.

Theorem 2. *For the 2-dimensional scheduling problem there is in general not a mechanism that is dominant strategy incentive compatible, individually rational and minimizes the expected total payment made to jobs, that achieves the same expected total payments as the optimal mechanism.*

Proof. Consider the following instance with two jobs. Job 1 has type space $W_1 \times P_1 = \{5,7\} \times \{3,7\}$ and job 2 has type space $W_2 \times P_2 = \{2,4,7\} \times \{5,6\}$. The corresponding probabilities for the jobs' types are

$$\begin{array}{llll} \phi_1(5,3) = 0.5 & \phi_1(5,7) = 0.3 & \phi_1(7,3) = 0.1 & \phi_1(7,7) = 0.1 \\ \phi_2(2,5) = 0.2 & \phi_2(2,6) = 0.0 & \phi_2(4,5) = 0.0 & \phi_2(4,6) = 0.5 \\ \phi_2(7,5) = 0.3 & \phi_2(7,6) = 0.0. & & \end{array}$$

We will show that for this instance there is no allocation rule that is dominant strategies implementable and achieves the same expected total payments as the optimal allocation rule.

The instance has 2 different schedules and as job 1 has four types and job 2 has six types, there are 24 different type profiles. We will denote each of these type profiles $(w_1, p_1); (w_2, p_2)$ by a different case.

$$\begin{array}{lll} \text{case } a = (5,3),(2,5) & \text{case } b = (5,3),(2,6) & \text{case } c = (5,3),(4,5) \\ \text{case } d = (5,3),(4,6) & \text{case } e = (5,3),(7,5) & \text{case } f = (5,3),(7,6) \\ \text{case } g = (5,7),(2,5) & \text{case } h = (5,7),(2,6) & \text{case } i = (5,7),(4,5) \\ \text{case } j = (5,7),(4,6) & \text{case } k = (5,7),(7,5) & \text{case } l = (5,7),(7,6) \\ \text{case } m = (7,3),(2,5) & \text{case } n = (7,3),(2,6) & \text{case } o = (7,3),(4,5) \\ \text{case } p = (7,3),(4,6) & \text{case } q = (7,3),(7,5) & \text{case } r = (7,3),(7,6) \\ \text{case } s = (7,7),(2,5) & \text{case } t = (7,7),(2,6) & \text{case } u = (7,7),(4,5) \\ \text{case } v = (7,7),(4,6) & \text{case } w = (7,7),(7,5) & \text{case } x = (7,7),(7,6) \end{array}$$

Let us first compute the payments made by the optimal mechanism. In total there are $2^{24} = 16.777.216$ allocation rules. For this instance the unique optimal allocation rule f assigns the following schedules[19] to each case.

$$\begin{aligned} &\text{case } a, b, d, g, h, m, n, o, p, q, s, t, u, v, w \to 12 \\ &\text{case } c, e, f, i, j, k, l, r, x \to 21 \end{aligned}$$

[19] As in Section 5.1, schedule 12 represents the schedule where job 1 is processed before job 2.

Computing the expected start times of job 1, the individual rationality constraints for this job state

$$\begin{aligned}
E\pi_1^f(5,3) &\geq 5 \cdot ES_1(f,5,3) = 5 \cdot 1.5 = 7.5 \\
E\pi_1^f(5,7) &\geq 5 \cdot ES_1(f,5,7) = 5 \cdot 4.5 = 22.5 \\
E\pi_1^f(7,3) &\geq 7 \cdot ES_1(f,7,3) = 7 \cdot 0.0 = 0 \\
E\pi_1^f(7,7) &\geq 7 \cdot ES_1(f,7,7) = 7 \cdot 0.0 = 0.
\end{aligned}$$

As in the previous section we iteratively regard the incentive constraints for which the right-hand side exceeds zero. It happens to be that already in the first step all incentive constraints are fulfilled. Therefore, for job 1 the payments are determined only by the individual rationality constraints. For job 2 the individual rationality constraints state

$$\begin{array}{lll}
E\pi_2^f(2,5) \geq 9.2 & E\pi_2^f(2,6) \geq 9.2 & E\pi_2^f(4,5) \geq 4.0 \\
E\pi_2^f(4,6) \geq 10.0 & E\pi_2^f(7,5) \geq 7.0 & E\pi_2^f(7,6) \geq 0.0.
\end{array}$$

The incentive constraints for which the right-hand side exceeds zero are

$$\begin{aligned}
E\pi_2^f(2,5) - 2 \cdot ES_2(f,2,5) &\geq E\pi_2^f(4,5) - 2 \cdot ES_2(f,4,5) \\
E\pi_2^f(2,5) - 2 \cdot ES_2(f,2,5) &\geq E\pi_2^f(4,6) - 2 \cdot ES_2(f,4,6) \\
E\pi_2^f(2,5) - 2 \cdot ES_2(f,2,5) &\geq E\pi_2^f(7,5) - 2 \cdot ES_2(f,7,5) \\
E\pi_2^f(2,6) - 2 \cdot ES_2(f,2,6) &\geq E\pi_2^f(4,6) - 2 \cdot ES_2(f,4,6) \\
E\pi_2^f(4,5) - 4 \cdot ES_2(f,4,5) &\geq E\pi_2^f(7,5) - 4 \cdot ES_2(f,7,5).
\end{aligned}$$

From this follows that $E\pi_2^f(2,5), E\pi_2^f(2,6) \geq 14.2$ and $E\pi_2^f(4,5) \geq 7$. Regarding anew the incentive constraints with right-hand side greater than zero, we notice that already in the second iteration, all incentive constraints are fulfilled. Therefore the expected total payment achieved by allocation rule f is

$$\begin{aligned}
EP^{min}(f) &= \sum_{t_1 \in T_1} \phi_1(t_1) E\pi_1^f(t_1) + \sum_{t_2 \in T_2} \phi_2(t_2) E\pi_2^f(t_2) \\
&= 0.5 \cdot 7.5 + 0.3 \cdot 22.5 + 0.1 \cdot 0 + 0.1 \cdot 0 + 0.2 \cdot 14.2 \\
&\quad + 0.0 \cdot 14.2 + 0.0 \cdot 7.0 + 0.5 \cdot 10.0 + 0.3 \cdot 7.0 + 0.0 \cdot 0.0 \\
&= 20.44.
\end{aligned}$$

Our second step is to find the allocation rule that is dominant strategy implementable, individually rational and minimizes the expected total

payments made to jobs. This happens to be allocation rule f, the same allocation rule that is optimal for the BNIC setting. As we consider a DSIC setting, we take into account individual rationality constraints (2.7) and incentive constraints (2.5). Apart from the fact that we consider the IR and DSIC constraints, we apply the same method as for the BNIC setting. Denote by $\pi_1^f(\alpha)$ and $\pi_2^f(\alpha)$ the payment to job 1 and job 2 for case α, under allocation rule f, respectively. For example, if $\alpha = (t_1, t_2)$, we denote by $\pi_1^f(\alpha)$ the payment to job 1, given that it reports type t_1 and job 2 reports type t_2. The individual rationality and incentive constraints for job 1 and job 2 imply

$$\begin{array}{llll}
\pi_1^f(c) \geq 25 & \pi_1^f(e) \geq 25 & \pi_1^f(f) \geq 42 & \pi_1^f(i) \geq 25 \\
\pi_1^f(j) \geq 30 & \pi_1^f(k) \geq 25 & \pi_1^f(l) \geq 42 & \pi_1^f(r) \geq 42 \\
\pi_1^f(x) \geq 42 & & & \\
\pi_2^f(a) \geq 12 & \pi_2^f(b) \geq 12 & \pi_2^f(d) \geq 12 & \pi_2^f(\mathrm{g}) \geq 14 \\
\pi_2^f(h) \geq 14 & \pi_2^f(m) \geq 21 & \pi_2^f(n) \geq 12 & \pi_2^f(o) \geq 21 \\
\pi_2^f(p) \geq 12 & \pi_2^f(q) \geq 21 & \pi_2^f(s) \geq 49 & \pi_2^f(t) \geq 28 \\
\pi_2^f(u) \geq 49 & \pi_2^f(v) \geq 28 & \pi_2^f(w) \geq 49 &
\end{array}$$

whereas all other payments equal 0. Using this payment scheme we can compute the expected payment to job j when it reports type t_j, in order to compare it to the expected payment in the BNIC setting[20]. We notice that all expected payments are similar, except for the expected payment to job 2 reporting type $(2, 5)$, which occurs in cases a, g, m, s. Let us denote by $(t_1, (2, 5))$ the case where job 1 reports type t_1 and job 2 reports type $(2, 5)$, then for the DSIC setting we have

$$\sum_{t_1 \in T_1} \phi_1(t_1) \cdot \pi_2^f(t_1, (2, 5)) = 0.5 \cdot 12 + 0.3 \cdot 14 + 0.1 \cdot 21 + 0.1 \cdot 49 = 17.2$$

while in the BNIC setting we have $E\pi_2^f(2, 5) = 14.2$. The raise of this expected payment by 3, compared to the BNIC case, should be solely responsible for the raise in expected total payments to the jobs in comparison to the BNIC setting. The minimal expected total payment achieved by allocation rule f for the DSIC setting can be computed by summing over all cases for all jobs and multiplying by the corresponding probability. Denoting

[20]The expected payment in de BNIC setting is under the assumption that all jobs other than job j report truthfully.

by $\phi(i)$ the probability on case, or equivalently, type profile i, we have

$$P^{min}(f) = \sum_{j \in J} \sum_{t_j \in T_j} \sum_{t_{-j} \in T_{-j}} \phi_j(t_j)\phi_{-j}(t_{-j})\pi_j^f(t_j, t_{-j}) = 21.04.$$

Note that as expected, $P^{min}(f) - EP^{min}(f) = 0.2 \cdot 3 = 0.6$, due to the raise in expected payment to job 2, reporting type $(2, 5)$, and the fact that $\phi_2(2, 5) = 0.2$. □

As for the previous section, there still might be specific instances for which BNIC-DSIC equivalence holds. However, instances where types of jobs have product distributions do not belong to that group.

Theorem 3. *For the 2-dimensional scheduling problem, where types of jobs have a product distribution, Bayes-Nash and dominant strategy incentive compatibility is not equivalent.*

Proof. Consider the following instance with two jobs. Job 1 has type space $W_1 \times P_1 = \{3, 6\} \times \{2, 5\}$ and job 2 has type space $W_2 \times P_2 = \{4, 5, 9\} \times \{1, 8\}$. Note that the probability distribution ϕ_j for the type of job j is a product distribution and therefore we have weight probabilities $\varphi_j(w_j)$ and processing time probabilities $\psi_j(p_j)$.

$$\begin{array}{llll} \varphi_1(3) = 0.4 & \varphi_1(6) = 0.6 & \psi_1(2) = 0.3 & \psi_1(5) = 0.7 \\ \varphi_2(4) = 0.4 & \varphi_2(5) = 0.2 & \varphi_2(9) = 0.4 & \psi_2(1) = 0.5 \\ \psi_2(8) = 0.5. & & & \end{array}$$

Note that from these probabilities follow the corresponding probabilities for types of jobs. We will show that for this instance there is no DSIC mechanism that is individually rational and achieves the same minimal expected total payments as the optimal mechanism.

Computing the minimal expected total payments for both the BNIC and DSIC setting, is similar to the previous example and therefore we omit most of the steps. The instance has 2 different schedules and 24 different type profiles, which we all denote by a different case.

case $a = (3, 2), (4, 1)$	case $b = (3, 2), (4, 8)$	case $c = (3, 2), (5, 1)$
case $d = (3, 2), (5, 8)$	case $e = (3, 2), (9, 1)$	case $f = (3, 2), (9, 8)$
case g $= (3, 5), (4, 1)$	case $h = (3, 5), (4, 8)$	case $i = (3, 5), (5, 1)$
case $j = (3, 5), (5, 8)$	case $k = (3, 5), (9, 1)$	case $l = (3, 5), (9, 8)$
case $m = (6, 2), (4, 1)$	case $n = (6, 2), (4, 8)$	case $o = (6, 2), (5, 1)$
case $p = (6, 2), (5, 8)$	case $q = (6, 2), (9, 1)$	case $r = (6, 2), (9, 8)$
case $s = (6, 5), (4, 1)$	case $t = (6, 5), (4, 8)$	case $u = (6, 5), (5, 1)$
case $v = (6, 5), (5, 8)$	case $w = (6, 5), (9, 1)$	case $x = (6, 5), (9, 8)$

For each of these cases we must compute the (expected) payments made to the jobs. Let us first compute the expected total payments made by the optimal mechanism. In total there are $2^{24} = 16.777.216$ allocation rules. It turns out that for this instance there are two optimal allocation rules. We will consider optimal allocation rule f, that assigns the following schedules to each case.

$$\begin{aligned}&\text{case } b, d, h, m, n, o, p, t, v \to 12\\ &\text{case } a, c, e, f, \mathrm{g}, i, j, k, l, q, r, s, u, w, x \to 21\end{aligned}$$

Considering individual rationality constraints (2.8) and incentive constraints (2.6) for both job 1 and job 2, we obtain lower bounds on the expected payments.

$$\begin{array}{lll} E\pi_1^f(3,2) \geq 12.60 & E\pi_1^f(3,5) \geq 15.00 & E\pi_1^f(6,2) \geq 10.80 \\ E\pi_1^f(3,2) \geq 12.60 & & \\ E\pi_2^f(4,1) \geq 4.14 & E\pi_2^f(4,8) \geq 19.10 & E\pi_2^f(5,1) \geq 4.14 \\ E\pi_2^f(5,8) \geq 13.50 & E\pi_2^f(9,1) \geq 0.90 & E\pi_2^f(9,8) \geq 0.00 \end{array}$$

This leaves us to compute the expected total payment achieved by allocation rule f, which is

$$\begin{aligned} EP^{min}(f) &= \sum_{t_1 \in T_1} \phi_1(t_1) E\pi_1^f(t_1) + \sum_{t_2 \in T_2} \phi_2(t_2) E\pi_2^f(t_2) \\ &= 0.12 \cdot 12.60 + 0.28 \cdot 15.00 + 0.18 \cdot 10.80 + 0.42 \cdot 12.60 \\ &\quad + 0.20 \cdot 4.14 + 0.20 \cdot 19.10 + 0.10 \cdot 4.14 + 0.10 \cdot 13.50 \\ &\quad + 0.20 \cdot 0.90 + 0.20 \cdot 0.00 \\ &= 19.54. \end{aligned}$$

Now we compute the payments achieved by the allocation rule that is dominant strategy implementable, individually rational and minimizes the expected total payments made to jobs. This is allocation rule g, that assigns the following schedules to each case.

$$\begin{aligned}&\text{case } b, d, h, m, n, o, p, r, t, v \to 12\\ &\text{case } a, c, e, f, \mathrm{g}, i, j, k, l, q, s, u, w, x \to 21\end{aligned}$$

As we consider a DSIC mechanism, we take into account individual rationality constraints (2.7) and incentive constraints (2.5). Combining these

constraints we compute for each case the payments to job 1 and job 2.

$$\begin{array}{llll}
\pi_1^g(a) \geq 6 & \pi_1^g(c) \geq 6 & \pi_1^g(e) \geq 6 & \pi_1^g(f) \geq 48 \\
\pi_1^g(\mathrm{g}) \geq 6 & \pi_1^g(i) \geq 6 & \pi_1^g(j) \geq 24 & \pi_1^g(k) \geq 6 \\
\pi_1^g(l) \geq 48 & \pi_1^g(q) \geq 6 & \pi_1^g(s) \geq 6 & \pi_1^g(u) \geq 6 \\
\pi_1^g(w) \geq 6 & \pi_1^g(x) \geq 48 & &
\end{array}$$

$$\begin{array}{llll}
\pi_2^g(a) \geq 2 & \pi_2^g(b) \geq 10 & \pi_2^g(c) \geq 2 & \pi_2^g(d) \geq 10 \\
\pi_2^g(e) \geq 2 & \pi_2^g(h) \geq 20 & \pi_2^g(m) \geq 18 & \pi_2^g(n) \geq 18 \\
\pi_2^g(o) \geq 18 & \pi_2^g(p) \geq 18 & \pi_2^g(r) \geq 18 & \pi_2^g(s) \geq 5 \\
\pi_2^g(t) \geq 25 & \pi_2^g(u) \geq 5 & \pi_2^g(v) \geq 25 & \pi_2^g(w) \geq 5
\end{array}$$

This finally gives us the minimal expected total payment for allocation rule

$$P^{min}(g) = \sum_{j \in J} \sum_{t_j \in T_j} \sum_{t_{-j} \in T_{-j}} \phi_j(t_j)\phi_{-j}(t_{-j})\pi_j^g(t_j, t_{-j}) = 19.612$$

which claims our proof. □

Although BNIC-DISC equivalence neither holds for instances where types of jobs have a product distribution, there might be other specific instances for which BNIC is equivalent to DSIC. This has to be revealed by further (computational) research.

6 Conclusion

We have considered a strategic scheduling problem analysed by Heydenreich et al, where a set of selfish, risk neutral jobs have to be processed non-preemptively on a single machine. For the 2-dimensional setting of this scheduling problem, both the weight and processing time of a job, referred to as a job's type, are private information. The service provider assigns a schedule to each possible report of types, and compensates jobs for their waiting time in the form of payments. For this setting we sought to get more insight into optimal mechanisms, i.e. incentive compatible mechanisms that minimize the expected total payments made to jobs.

In the flavour of recent work on automated mechanism design as proposed by Conitzer and Sandholm, we have proposed a mathematical programming formulation for the 2-dimensional optimal mechanism design problem. Although the corresponding mathematical programs are too extensive to solve for real-world instances, they are well suited for solving smaller instances up to four jobs to test and generate hypotheses.

Systematically generating small instances at random, we constructed MPs for different types of mechanisms in the mechanism design problem, which we subsequently solved using ILOG CPLEX. This way, we found an example to prove that the optimal allocation rule in general does not satisfy a condition called independence of irrelevant alternatives (IIA). This gives a hint towards intractability of the 2-dimensional optimal mechanism design problem. However, despite extensive computational research, we could not find a similar result for instances where jobtypes have a product distribution. This leads to the conjecture that for these specific instances the optimal allocation rule does satisfy the IIA condition. Therefore, for instances where jobtypes have a product distribution, possibly there exists an explicit formula for the optimal payment scheme.

Moreover we have proposed an example to prove that Bayes-Nash incentive compatibility and dominant strategy incentive compatibility is in general not equivalent for the 2-dimensional scheduling problem. That is, in general there is not a DSIC mechanisms that achieves the same expected total payment as the optimal Bayes-Nash mechanism. Even more, BNIC-DSIC equivalence does not even hold for instances where jobtypes have a product distribution.

7 Future Research

Although we have answered a number of questions around the 2-dimensional single machine scheduling problem, many remain open. There are several directions for future research which can be taken to further analyse the 2-dimensional scheduling problem.

A possible direction is to do more computational research using the mathematical programs we proposed in this thesis. Concerning optimal allocation rules satisfying the IIA condition, one could do more computational research on instances where jobtypes have a product distribution. Furthermore one could investigate other specific instances, for which the optimal allocation rule might satisfy the IIA condition. For example, one could investigate if the optimal allocation rule that is implementable in dominant strategies and achieves the minimal expected total payments, satisfies the IIA condition. For BNIC-DSIC equivalence likewise, there can be searched for specific instances for which BNIC-DSIC equivalence does hold.

In order to do more computational research, it would be convenient to improve the MPs such that the solving time by ILOG CPLEX decreases. Presumably, further reducing the number of constraints or variables, or reducing symmetry will not drastically change the solving time. However, a completely different mathematical programming formulation might lead to significant changes. One can think of a formulation where we do not define a schedule σ by explicitly assigning a job to a position, but by choosing the relative order of jobs. We would define binary variables

$$d_{ij}(t) = \begin{cases} 1 & \text{if job } i \text{ is processed before job } j \text{ given type profile } t \\ 0 & \text{otherwise} \end{cases}$$

which together with some additional constraints encode a feasible allocation rule. The advantage of this formulation is that we can very easily impose IIA by defining binary variables

$$d_{ij}(t_i, t_j) = \begin{cases} 1 & \text{if job } i \text{ is processed before job } j \text{ given type } t_i \text{ and } t_j \\ 0 & \text{otherwise} \end{cases}$$

instead of variables $d_{ij}(t)$. Now, by definition, the relative order of jobs i and j does not depend on the reported types of other jobs. Possibly, the solving time of the MPs which are constructed using this new formulation, reduces substantially. This would allow one to solve more instances in the same amount of time, or to solve instances with an increased number of jobs or jobtypes. Important is to note, that as the solving time by ILOG CPLEX

is being decreased, the time it takes C++ to build a LP file containing the MP for a given setting, becomes a more substantial. However, this step of the C++ program has not been optimized, and therefore we expect that the time it takes to build a LP file can be decreased easily.

Finally one could choose to leave the direction of computational research and switch to a more manual approach. By analysing the individual rationality and incentive compatibility definitions very carefully, one could try to find by hand, specific instances for which the optimal allocation rule satisfies the IIA condition or BNIC-DSIC equivalence holds. However, one could also, using mathematical programming formulations, analyse the effect of certain properties of instances on the expected total payments. In this way one might generate hypotheses backed by empirical evidence, which afterwards can be subjected to theoretical analysis.

References

[1] Nobel Prize Committee. Leonid hurwicz, eric s. maskin and roger b. myerson: Mechanism design theory. Nobel Prize in Economics documents 2007-2, Nobel Prize Committee, October 2007.

[2] Vincent Conitzer and Tuomas Sandholm. Complexity of mechanism design. *CoRR*, cs.GT/0205075, 2002.

[3] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms*. MIT Press, 2009.

[4] Alex Gershkov, Benny Moldovanu, and Xianwen Shi. Bayesian and dominant strategy implementation revisited. Working Papers tecipa-422, University of Toronto, Department of Economics, February 2011.

[5] B. Heydenreich, D. Mishra, R. Müller, and M.J. Uetz. Optimal mechanisms for single machine scheduling. In C. Papadimitriou and S. Zhang, editors, *Internet And Network Economics (WINE 2008)*, volume 5385 of *Lecture Notes in Computer Science*, pages 414–425, Berlin, December 2008. Springer Verlag.

[6] B. Heydenreich, D. Mishra, R. Müller, and M.J. Uetz. Optimal mechanisms for scheduling. In Susanne Albers, Sanjoy K. Baruah, Rolf H. Möhring, and Kirk Pruhs, editors, *Scheduling*, number 10071 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.

[7] L. V. Kantorovich. Mathematical Methods of Organizing and Planning Production. *Management Science*, 6(4):366–422, 1960.

[8] Alejandro M. Manelli and Daniel R. Vincent. Bayesian and Dominant-Strategy Implementation in the Independent Private-Values Model. *Econometrica*, 78(6):1905–1938, 2010.

[9] A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic theory*. Oxford University Press, 1995.

[10] Roger B. Myerson. Optimal Auction Design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[11] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[12] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[13] Tuomas Sandholm. Automated mechanism design: A new application area for search algorithms. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP 03), Kinsale, County*. Springer, 2003.

[14] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

A Symbols and Abbreviations

Symbol	Description
J	Set of jobs $\{1, \ldots, n\}$
w_j	True weight of job j
$\tilde{w}_j$	Reported weight of job j
W_j	Weight-set of job j, i.e. $\{w_1^1, \ldots, w_j^{m_j}\}$
w_{-j}	True weight vector for all jobs except job j
$\tilde{w}_{-j}$	Reported weight vector for all jobs except job j
W_{-j}	Set of all w_{-j}
w	Weight vector $(w_1, \cdots, w_n)$
W	Set of all weight vectors
p_j	True processing time of job j
$\tilde{p}_j$	Reported processing time of job j
P_j	Processing time-set of job j, i.e. $\{p_1^1, \ldots, p_j^{q_j}\}$
t_j	True type of job j
$\tilde{t}_j$	Reported type of job j
T_j	Type-set of job j
ϕ_j	Probability distribution for jobtypes of job j
t_{-j}	True type vector for all jobs except job j
$\tilde{t}_{-j}$	Reported type vector for all jobs except job j
T_{-j}	Set of all t_{-j}
ϕ_{-j}	Probability distribution associated with t_{-j}
t	Type profile, i.e. $(t_1, \ldots, t_n)$
T	Set of all type profiles
ϕ	Joint probability distribution of w
φ_j	Probability distribution associated with w_j (only when ϕ is a product distribution)
ψ_j	Probability distribution associated with p_j(only when ϕ is a product distribution)
σ	Schedule in which jobs are order on machine
$\mathfrak{S}$	Set of all schedules
σ_j	Position of job j in schedule σ
$S_j(\sigma)$	Start time of job j given schedule σ
f	Allocation rule
π	Payment scheme
(f, π)	Mechanism
π_j	Payment to job j in payment scheme π

Symbol	Description
$-w_j S_j(\sigma)$	Valuation of job j for schedule σ when its true weight is w_j
$\pi_j - w_j S_j(\sigma)$	Utility of job j for schedule σ when its true weight is w_j and it additionally receives payment π_j
$ES_j(f, \tilde{t}_j)$	Expected start time of job j, when allocation rule f is applied and it reports type t_j
$E\pi_j(\tilde{t}_j)$	Expected payment to job j when it reports type t_j
$\pi^f(\cdot)$	Payment scheme that minimizes the expected total payments made to the jobs, when allocation rule f is applied (DSIC setting)
$\pi_j^f(\tilde{t}_j, t_{-j})$	Payment made to job j when payment scheme π^f is applied, it reports type $\tilde{t}_j$ and the other jobtypes are t_{-j} (DSIC setting)
$E\pi^f(\cdot)$	Payment scheme that minimizes the expected total payments made to the jobs, when allocation rule f is applied (BNIC setting)
$E\pi^f(\tilde{t}_j)$	Payment made to job j when payment scheme π^f is applied and it reports type $\tilde{t}_j$ (BNIC setting)
$P^{min}(f)$	Minimal expected total payment made to the jobs when allocation rule f is applied (DSIC setting)
$EP^{min}(f)$	Minimal expected total payment made to the jobs when allocation rule f is applied (BNIC setting)

Abbreviation	Description
$BNIC$	Bayes-Nash incentive compatible
$DSIC$	Dominant strategy incentive compatible
IR	Individually rational (DSIC setting)
IRE	Individually rational (BNIC setting)
IIA	Independence of irrelevant alternatives
MP	Mathematical program
MIP	Mixed integer program
$MIQCP$	Mixed inter quadratically constrained program

B Instance File Format

The text file shown below corresponds to the instance shown in Figure 1, which can be found in Section 4.1.

```
3
1  1  4
2  7  1
8  4  1
2  3  0.1
2  8  0.1
5  3  0.6
5  8  0.2
```

C CPLEX LP file

The LP file shown below corresponds to the instance shown in Figure 1 and in Appendix B.

```
Minimize
 obj: Pi_0_0 + Pi_1_0 + 0.4 Pi_2_0 + 0.6 Pi_2_3
Subject To
 c1:  x_0_0 + x_0_1 + x_0_2 + x_0_3 + x_0_4 + x_0_5 = 1
 c2:  x_1_0 + x_1_1 + x_1_2 + x_1_3 + x_1_4 + x_1_5 = 1
 c3:  x_2_0 + x_2_1 + x_2_2 + x_2_3 + x_2_4 + x_2_5 = 1
 c4:  x_3_0 + x_3_1 + x_3_2 + x_3_3 + x_3_4 + x_3_5 = 1
 c5:  - 4 x_0_2 - 5.6 x_0_3 - 1.6 x_0_4 - 5.6 x_0_5 - 6
    x_3_2 - 8.4 x_3_3 - 2.4 x_3_4 - 8.4 x_3_5 + S_0_0 = 0
 c6:  - 4 x_0_0 - 5.6 x_0_1 - 5.6 x_0_4 - 1.6 x_0_5 - 6
    x_3_0 - 8.4 x_3_1 - 8.4 x_3_4 - 2.4 x_3_5 + S_1_0 = 0
 c7:  - 20 x_0_0 - 10 x_0_1 - 20 x_0_2 - 10 x_0_3 + S_2_0 =
    0
 c8:  - 20 x_1_0 - 10 x_1_1 - 20 x_1_2 - 10 x_1_3 + S_2_1 =
    0
 c9:  - 20 x_2_0 - 10 x_2_1 - 20 x_2_2 - 10 x_2_3 + S_2_2 =
    0
 c10: - 20 x_3_0 - 10 x_3_1 - 20 x_3_2 - 10 x_3_3 + S_2_3 =
    0
 c11: Pi_0_0 - 10 S_0_0 >= 0
 c12: Pi_1_0 - 10 S_1_0 >= 0
 c13: Pi_2_0 - 7 S_2_0 >= 0
 c14: Pi_2_1 - 4 S_2_1 >= 0
 c15: Pi_2_2 - 9 S_2_2 >= 0
 c16: Pi_2_3 - S_2_3 >= 0
 c17: Pi_2_0 - Pi_2_1 - 7 S_2_0 + 7 S_2_1 >= 0
 c18: Pi_2_0 - Pi_2_2 - 7 S_2_0 + 7 S_2_2 >= 0
 c19: Pi_2_0 - Pi_2_3 - 7 S_2_0 + 7 S_2_3 >= 0
 c20: - Pi_2_0 + Pi_2_2 + 9 S_2_0 - 9 S_2_2 >= 0
 c21: - Pi_2_1 + Pi_2_2 + 9 S_2_1 - 9 S_2_2 >= 0
 c22: Pi_2_2 - Pi_2_3 - 9 S_2_2 + 9 S_2_3 >= 0
 c23: - Pi_2_0 + Pi_2_3 + S_2_0 - S_2_3 >= 0
 c24: - Pi_2_1 + Pi_2_3 + S_2_1 - S_2_3 >= 0
 c25: - Pi_2_2 + Pi_2_3 + S_2_2 - S_2_3 >= 0
Bounds
 0 <= x_0_0 <= 1
 0 <= x_0_1 <= 1
 0 <= x_0_2 <= 1
 0 <= x_0_3 <= 1
 0 <= x_0_4 <= 1
```

```
 0 <= x_0_5 <= 1
 0 <= x_1_0 <= 1
 0 <= x_1_1 <= 1
 0 <= x_1_2 <= 1
 0 <= x_1_3 <= 1
 0 <= x_1_4 <= 1
 0 <= x_1_5 <= 1
 0 <= x_2_0 <= 1
 0 <= x_2_1 <= 1
 0 <= x_2_2 <= 1
 0 <= x_2_3 <= 1
 0 <= x_2_4 <= 1
 0 <= x_2_5 <= 1
 0 <= x_3_0 <= 1
 0 <= x_3_1 <= 1
 0 <= x_3_2 <= 1
 0 <= x_3_3 <= 1
 0 <= x_3_4 <= 1
 0 <= x_3_5 <= 1
Binaries
 x_0_0   x_0_1   x_0_2   x_0_3   x_0_4   x_0_5   x_1_0   x_1_1
     x_1_2   x_1_3   x_1_4   x_1_5   x_2_0   x_2_1   x_2_2   x_2_3
      x_2_4   x_2_5   x_3_0   x_3_1   x_3_2   x_3_3   x_3_4   x_3_5
End
```

D Pseudo-Code C++ Program

Algorithm 1: The C++ program realising all steps of the solution method

```
Input: Set of tasks and processors
Output: Mapping of tasks to processors
maxcnt = #counterexamples to be found;
cnt = 0;
while cnt < maxcnt do
    begin generate instance
        read/generate # jobs;
        for i ≤ # jobs do
            read/generate # types per job;
            for j ≤ # types per job do
                generate w_j, p_j, φ_j;
            end
        end
        write instance to text file;
    end
    begin construct MPs
        read types of mechanisms S_1 and S_2 to be compared;
        read instance text file ;
        for S_1, S_2 do
            Construct MP ;
            Write MP to CPLEX LP file;
        end
    end
    begin Compare MPs
        opt_1 = solution MP for S_1 using ILOG CPLEX;
        opt_2 = solution MP for S_2 using ILOG CPLEX;
        if opt_1 ≠ opt_2 then
            Save instance;
            Save CPLEX LP file for S_1, S_2;
            cnt++;
        end
    end
end
```