Game-theoretic analysis of networks: Designing mechanisms for scheduling

Angelina Vidali[1]

Department of Informatics
University of Athens

Advisor: Elias Koutsoupias

August 17, 2009

[1]Supported in part by IST-15964 (AEOLUS), IST-2008-215270 (FRONTS), and the Greek GSRT

Abstract

Algorithmic mechanism design is an important area between computer science and economics. One of the most fundamental problems in this area is the problem of scheduling unrelated machines to minimize the makespan. The machines behave like selfish players: they have to get paid in order to process the tasks, and would lie about their processing times if they could increase their utility in this way. The problem was proposed and studied in the seminal paper of Nisan and Ronen, where it was shown that the approximation ratio of mechanisms is between 2 and n.

In this thesis, we present some recent improvements of the lower bound to $1+\sqrt{2}$ for three or more machines and to $1+\varphi$ for many machines. Since the gap between the lower bound of 2.618 and the upper bound of n is huge, we also propose an alternative approach to the problem, which first attempts to characterize all truthful mechanisms and then study their approximation ratio. Towards this goal, we show that the class of truthful mechanisms for two players (regardless of approximation ratio) is very limited: tasks can be partitioned in groups allocated by affine minimizers (a natural generalization of the well-known VCG mechanism) and groups allocated by threshold mechanisms.

Finally we generalize a tool we have used in the proof of the $1+\sqrt{2}$ lower bound: we give a geometrical characterization of truthfulness for the case of three tasks, which we believe that might be useful for proving improved lower bounds and which provides a more complete understanding of truthfulness.

SUBJECT AREA: Algorithms

KEYWORDS: algorithmic game theory, mechanism design, lower bound, characterization approximation algorithms

To my parents, to my brother and sister and to anyone who will happen to read after this page.

Acknowledgements

I first got to know my advisor Elias Koutsoupias while attending his lesson in Algorithms and Complexity and his highly motivating lectures with chalk on the blackboard were very different from what I had known until then. Big parts of the proof were missing –nobody would have noticed if it wasn't for the statement "I am now cheating."– and on the blackboard was just the idea. The best algorithm would only get revealed after having tried and rejected many other solutions. It was like solving an open problem: you try and believe for a while ideas and ways that might not lead anywhere, start from scratch many times before finding a solution. A very lonely procedure in math had been turned into a course. I would like to thank him for the freedom he trusted me, for insisting on simple and clear ideas, solutions and statements, and for his demand that I should never compromise myself with anything mediocre.

I would also like to thank:

Christos Athanasiadis and Ioannis Emiris for helpful discussions concerning the geometrical part of my thesis and by brother Aris for drawing the nice 3D models of the complicated partitions of the space I include in Chapter 3. I would additionally like to thank Ioannis Emiris for many interesting conversations in algebra and geometry and a Course in Computational Geometry that I really enjoyed.

The members of the my PhD Committee: Vassilis Zissimopoulos for his interest and readiness to help with any problem, and Stavros Kollipoulos for his guidance to the classical version of the scheduling unrelated machines problem.

Paul Spirakis for many interesting, motivating and empowering discussions and also for the very fertile research environment of CTI: some people from CTI I would like to thank especially for many interesting discussions are Ioannis Caragiannis, Spyros Kontogiannis and Panagiota Panagopoulou.

Stathis Zachos for his brainteasers, (which sometimes were open problems!), for

the warm and friendly academic environment of Corelab that he has created and the talks, meetings and interesting discussions I have had during my frequent visits there with: Costis Georgiou, Panos Cheilaris, Costis Daskalakis, Dimitris Fotakis, Georgia Kaouri, Nikos Leonardos, Vangelis Markakis, Vaggelis Mpampas, Aris Pagourtzis, Vassilis Syrkganis and many others.

My master's advisor Yiannis Moschovakis for the guidance and trust he showed me and for still urging me to become better than myself.

My friends, fellow-students and collaborators Giorgos Christodoulou, Yiannis Giannakopoulos, Katia Papakonstantinopoulou, George Pierrakos, Manolis Pountourakis as well as my officemates Antonis, Chrisda, Christos, George, Vassilis for our wonderful collaboration and the time we spent together.

Monika Henzinger and Monaldo Mastrollili for their invitations to EPFL and IDSIA respectively and for the very interesting discussions I had there. I would also like to thank Friedhelm Meyer auf der Heide for inviting me to Paderborn and Amos Fiat for the very interesting discussions we had during his visit in Athens.

Finally I am grateful for the GSRT scholarship that funded my PhD studies as well as for the scholarships I have recieved during my studies from the Alexandros Onassis Foundation, IKY, Antonis Papadakis Fund, ASL as well as the experience I gained from the research projects AEOLUS and FRONTS.

Contents

1 Social Choice **3**

1.1 Designing a Mechanism 3
1.2 Different Domains of valuations 5
1.3 From Valuations to Types 6
1.4 The Vickrey auction 8
1.5 Properties of truthful mechanisms 9
1.6 The VCG mechanism 9
1.7 Implementation in Dominant Strategies 11
1.8 The Revelation Principle 12
1.9 Outline of the Thesis 13

2 An introduction to the Scheduling Problem **15**

2.1 Statement of the problem 15
2.2 An n-approximate truthful mechanism 17
2.3 Monotonicity: A local characterization of Truthfulness 19
2.4 Known mechanisms for the Scheduling Problem 25
2.5 Additive and threshold mechanisms 28
2.6 Some easy lower bounds 28
2.7 The allocation graph of each player 32
2.8 Related work 35

3 The geometry of truthfulness **37**

3.1 The problem 37
3.1.1 Our Tools 39
3.2 The example of two tasks demonstrates the idea 40
3.3 Definitions 40

3.4 Characterization of 3-Dimensional mechanisms 43
3.4.1 Calculating the distances 43
3.4.2 Properties satisfied by the allocation graph 44
3.4.3 All possible Mechanisms 46
3.4.4 Knowing a few distances we can draw the whole mechanism . 46
3.5 Lower bounds for some Scheduling Mechanisms 48
3.6 Concluding Remarks and open problems 50
3.7 Some more figures 50

4 A lower bound of $1+\sqrt{2}$ 55
4.1 A geometrical lemma 55
4.2 The proof of the main result 56
4.3 Lower bound for other L_p norms 60

5 A lower bound of $1+\varphi$ 63
5.1 A lower bound of $1+\varphi$ for $n \to \infty$ machines. 63

6 Characterization of 2-player mechanisms 69
6.1 Introduction 69
6.1.1 Do we need the full power of a characterization to get a lower bound? 69
6.1.2 What kind of characterization? 70
6.2 The characterization of decisive mechanisms for 2 tasks 75
6.2.1 Using a "fix and release" Characterization 75
6.2.2 Characterization of mechanisms with simple boundaries. . . . 78
6.2.3 Continuous boundaries can only be univariate functions 87
6.2.4 Extending the characterization to non-continuous functions . . 90
6.2.5 Transforming $f_{a:a'}$ to a univariate function $f'_{a:a'}$. 92
6.2.6 Applying the inverse transformation 93
6.3 The case of many tasks 95
6.4 Lower bound for 2 tasks 97
6.5 Concluding remarks 99

Chapter 1

Social Choice

> "Social choice characterization theorems describe functions which map from a collection of preferences, views or welfares in various states to a single preference, view or welfare over the states."
>
> Kevin Roberts

1.1 Designing a Mechanism

A social choice is a single joint decision which is made after taking into account the preferences of different individuals affected by the decision. The most common examples of social choice are voting, auctions and government policy. Mechanism design takes into account the selfish strategic behavior of the single individuals (in a game theoretic sense) in order to design an algorithm or protocol that makes this social choice. The reason we need mechanism design is that the preferences of the individuals are private. Think that most people keep their political preferences to themselves and that very often people who really favor a minor party candidate will vote for the least undesirable from the parties that are most likely to take the government rather than "throw away their vote".

In the social choice setting, there are n players and a set of possible outcomes $\mathcal{A}$. For example when voting for a winner the possible outcomes are the competing parties, in auctions an allocation of the items for sale to the players.

Each player has a valuation function $v_i : \mathcal{A} \to \mathbb{R}$ which gives the value of player i for every outcome. We denote by $V_i \subseteq \mathbb{R}^{\mathcal{A}}$ the set of possible valuation functions for

player i. The valuation function maps different outcomes to different real numbers. This is like making the assumption that preferences can be measured using money. You can think of the real number corresponding to an outcome as the quantity of "money" the player gains if this outcome is chosen.

The most important reason we need to represent preferences with real values is that this allows us to use payments. Even though this assumption leaves room for discussion it is fairly reasonable in many cases. Additionally mechanism design without money faces very strong impossibility theorems like the Gibbard-Satterthwaite theorem for voting systems, which states that if there are more than two candidates all non-dictatorial voting rules are manipulable (in the sense that there exist situations in which a voter would benefit from reporting its preferences insincerely). The payments are a tool in the hands of the mechanism designer, in the sense that some social goals are impossible to realize without the use of payments.

The goal of each player is certainly to maximize his utility. But how does a player's payment p_i combine with his valuation to give the player's utility, that is how do payments affect the preferences of a player? We assume (as is common in most of the literature in economics) that the utility functions are *quasi-linear*, that is they are of the form $u_i(a_i, p_i) = v_i(a_i) - p_i$. They are called so because the dependence on money is separable from the valuation in a linear way.

A mechanism needs to choose an alternative and specify payments. For each n-tuple of valuations $v = (v_1, \ldots, v_n)$ the mechanism chooses an outcome $a(v)$ from the set of possible outcomes $\mathcal{A}$.

Definition 1 (Direct Revelation Mechanism)**.** A direct revelation mechanism consists of a social choice function $a : V_1 \times \ldots \times V_n \to \mathcal{A}$ and a vector of payment functions $p = (p_1, \ldots, p_n)$ such that $p_i : V_1 \times \ldots \times V_n \to \mathbb{R}$.

By the term "direct revelation" we mean that the mechanism takes as input the true valuations of the players (and not some lying strategy). However we still need a way to devise rules for the game in such a way that individuals will express their true tastes even when they act rationally. The goal of the mechanism designer is to choose a single outcome $a(v)$ that amalgamates the preferences v_i of the players. This means that $a(v)$ is such that even a player who would strictly prefer some other outcome instead of $a(v)$, cannot impose an outcome more favorable for him by changing his declaration and choosing a different strategy.

We next define the notion of truthfulness also known as incentive compatibility, or strategyproofness. By lying a player can change the allocation given to him and thus change his payment and utility. A mechanism is truthful if the players cannot gain anything by lying. So if a player with valuation v_i reports to the mechanism that his valuation is v_i' his utility does not increase.

Let $v = (v_1, \ldots, v_n)$ be an n-dimensional vector we denote (as is standard in game theory) by $v_{-i} := (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ the $(n-1)$-dimensional vector that we get from v when we remove the i-th coordinate. According to this notation we usually write (v_i, v_{-i}) instead of v to indicate the game theoretic situation player i faces.

Definition 2 (truthfulness)**.** A mechanism $(a, p_1, \ldots, p_n)$ is called truthful if for every player i and every possible n-tuple of valuations $v = (v_1, \ldots, v_n)$ and every $v_i' \in V_i$ we have

$$v_i(a(v_i, v_{-i})) - p_i(v_i, v_{-i}) \geq v_i(a(v_i', v_{-i})) - p_i(v_i', v_{-i}).$$

In Section 1.8 we will see that concentrating on truthful mechanisms is equivalent to concentrating on mechanism that have dominant equilibria.

1.2 Different Domains of valuations

The unrestricted domain of valuations is the case when $V_i = \mathbb{R}^{\mathcal{A}}$, that is, the valuations can be any real functions. The unrestricted domain of valuations is not a very natural domain because usually the valuations for two different outcomes might depend on each other and in this sense the players are more predictable. Imposing restrictions on the domain of valuations usually makes things more realistic, but also leaves room for the mechanism designer, in the sense that the set of possible truthful mechanisms is potentially richer.

Some of the most common restrictions we impose on valuations, most usually when studying different kinds of auctions, are the following: Suppose that the mechanism chooses an allocation $A = (A_1, \ldots, A_n)$ where A_i is the set of items allocated to player i and $A_i \cap A_j = \emptyset$ for $i \neq j$ we often assume the following conditions:

No externalities. The valuation v_i of player i should only depend on i's allocated bundle A_i, i.e. $v_i(A) = v_i(A_i)$.

Free Disposal. The valuation should be non-decreasing with the set of allocated items, i.e. for every $A_i \subseteq B_i$ we have that $v_i(A_i) \leq v_i(B_i)$.

Normalization. When a player gets nothing his valuation is zero, i.e. $v_i(\emptyset) = 0$.

The combinatorial auction domain [34]. The domain we get if we demand that v_i satisfies the preceding three conditions: no externalities, free disposal and normalization.

Special subdomains result when we impose additional restrictions:

Sub-additive valuations. A valuation v_i is subadditive if for any two sets A_i and A_i', $v_i(A_i) + v_i(A_i') \leq v_i(A_i \cup A_i')$.

Superadditive valuations. For any two disjoint sets A_i and A_i', $v_i(A_i) + v_i(A_i') \geq v_i(A_i \cap A_i')$.

Submodular valuations. An important special case of subadditive functions are sub- modular functions. A valuation function v_i is submodular if for any two sets A_i and A_i', $v_i(A_i) + v_i(A_i') \geq v_i(A_i \cup A_i') + v_i(A_i \cap A_i')$.

Remark 1. Here we denote by A_i the set of items allocated to player i, while in the rest of this work we denote by a_i the binary vector where a_{ij} is 1 if player i gets item j and 0 if he doesn't. It is however very easy to go from one notation to the other:

$$a_{ij} = \begin{cases} 1 & \text{if } j \in A_i \\ 0 & \text{else.} \end{cases}$$

1.3 From Valuations to Types

We have modeled the players as individuals who choose a valuation function from a known set of functions (for example if the set of valuation functions contains all functions that satisfy additivity, a player can choose to report any function from this set) the expected preferences of an individual for different outcomes.

However in most real-life situations for which we need Mechanism Design, for example in voting, in public projects or in auctions it is rather absurd to ask the individuals involved to report a function that depicts their preferences. You just ask them to rank the candidates, or to give a score for each one of the candidates, to say how much they are determined to pay for the construction of a public project (like a road or a bridge), or how much they are determined to pay for getting a bundle of goods. So the input the mechanism asks from player i is a vector of real values t_i.

In fact knowing this input t_i one can then deduce the valuation function of player i. The vector t_i is called the *type* of player i. We denote by T_i the set of all possible types for player i. We call $D := T_1 \times \ldots \times T_n$ the *domain* of the mechanism design problem.

If a player tells the truth he reports his type vector t_i, otherwise he follows a lying strategy. By lying a player can change his allocation and payment. However the utility of the player does not only depend on the false type he declares but also on his true type, which gives his valuation. (We assume here that the mechanism designer cannot verify this after all jobs are finished and change the payments. For mechanisms with verification see the second part of [42].)

Example 1 (Single-item auction). For example in a single-item auction each player reports a bid t_i for the item in sale. The valuation fuction corresponding to type t_i is $v_i(a_i) = a_i t_i$, where $a_i \in \{0, 1\}$ is the allocation of player i, that is the valuation of player i is t_i if he gets the item and 0 otherwise.

Example 2 (The Scheduling Problem). In the scheduling problem there are m tasks and we wish to allocate them to n players/machines. The possible outcomes are the $n \times m$ matrices a such that $a_{ij} \in \{0, 1\}$ and $\sum_{j=1}^{n} a_{ij} = 1$, in other words each task has to be allocated to exactly one player. Each player reports a vector $t_i = (t_{i1}, \ldots, t_{in})$, his processing times for each one of the m tasks, that is t_{ij} is equal to the valuation $v_i(e_j)$ of player i for getting only task j. (We use the common notation for the j-th unitary vector $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is at position j.) We assume that the processing time/valuation of a machine that gets two tasks is the sum of the processing times for each one of the tasks separately. This is equivalent to saying that the valuation of the machine is additive. Consequently in the scheduling problem the valuation v_i of player i for the outcome a is given by $v_i(a) = \sum_{j=1}^{m} a_{ij} v_i(e_j) = \sum_{j=1}^{m} a_{ij} t_{ij}$.

Example 3 (The Fractional version of the Scheduling Problem). This is exactly like the scheduling problem, except that we allow a task to be split between two or more players. Consequently the possible outcomes are the $n \times m$ matrices a such that $a_{ij} \in \mathbb{R}^+$ and $\sum_{j=1}^{n} a_{ij} = 1$. Even though the possible outcomes are infinitely many we can deduce which is the valuation function of a player using again the vector $t_i = (t_{i1}, \ldots, t_{in})$ of processing times for each one of the m tasks.

Example 4 (Combinatorial Auctions). In the combinatorial auctions problem there

are m items and we wish to allocate them to n players/bidders. The possible outcomes are the same as in the scheduling problem. Each player reports a vector t_i, his valuation for each one of his possible allocations a_i. However here the valuations do not need to be additive.

As you might have already noticed from the previous examples when using types we say that each player has a single valuation $v_i(a_i, t_i)$ where $v_i : \mathcal{A} \times T_i \to \mathbb{R}$, while in Subsection 1.1 we defined valuations as functions $v_i : \mathcal{A} \to \mathbb{R}$ that just take as input an allocation. The connection between these two definitions is that if we fix a type t_i then $v_i(a_i, t_i)$ is a function from $\mathcal{A} \to \mathbb{R}$. If we fix two different types t_i, t_i' then defining $f_i(a_i) := v_i(a_i, t_i), f_i'(a_i) := v_i(a_i, t_i')$ we get two different valuation functions f_i, f_i' according to the notation in Subsection 1.1. According to this notation a player instead of reporting a type, i.e. his preferences for a finite number of outcomes, reports a function from a set of functions known from before to the mechanism designer. When getting types as input the mechanism designer, knowing again which is the set of possible valuation functions, uses these types in order to deduce the valuation functions.

1.4 The Vickrey auction

The most typical example of mechanism design is an auction for selling a single item. It describes a real-life problem, is very simple and embodies some of the basic ideas and limitations of mechanism design.

The setting is the following: We have a single item to sell and there are many bidders. The mechanism designer announces the mechanism and then the bidders submit their bids for the item in sealed envelopes. The mechanism computes which one of the bidders gets the item (allocation) and at what price (payment). The goal of the mechanism designer is to give the item to the bidder who values it highest (i.e. to maximize the social welfare).

Perhaps the most straightforward solution would be to give the item to the player with the maximum bid and charge him his bid. This allocation is however not truthful: Suppose that there are three players with types $(t_1, t_2, t_3) = (1, 2, 7)$. This mechanism would give the item to the one who bids 7 and charge him 7. However this player would increase his utility by lying and giving a lower bid. He could bid slightly higher than the second best bid, here $2 + \epsilon$ for some small $\epsilon > 0$:

He would still get the item and increase his utility from 0 ($v_3 - p_3 = 7 - 7$) to $5 - \epsilon$ ($v_3 - p'_3 = 7 - 2 - \epsilon$).

How could we change this mechanism to a truthful one? Suppose we keep the allocation part as it is. Can we change the payment and give the winner incentive to report his true valuation? The answer is yes and the resulting auction is the celebrated Vickrey auction. The winner pays the second-best bid, so that the player with the highest bid (here $t_3 = 7$) would have no reason to lie about his valuation, no matter if the other players tell the truth or not.

1.5 Properties of truthful mechanisms

Lemma 1 ([34])**.** *The price $p_i(t)$ of a truthful mechanism does not depend on the declaration t_i of player i, but only on his allocation $a_i(t)$ and the declarations of the other players, that is $p_i(t) = p_i(a_i(t), t_{-i})$.*

Proof. Suppose towards a contradiction that there exist t_i, t'_i such that $a_i(t_i, t_{-i}) = a_i(t'_i, t_{-i})$, but $p_i(t_i, t_{-i}) < p_i(t'_i, t_{-i})$. Then when the true processing times of player i are t_i he has incentive to declare falsely that his processing times are t'_i. His valuation remains the same (as we assumed that $a_i(t_i, t_{-i}) = a_i(t'_i, t_{-i})$) and his payment decreases. Consequently by declaring falsely t'_i his utility increases $v_i(a_i(t_i, t_{-i}), t_i) - p_i(t_i, t_{-i}) > v_i(a_i(t'_i, t_{-i}), t_i) - p_i(t'_i, t_{-i})$. This contradicts the assumption that the mechanism is truthful. □

Theorem 1. *For every player i and type t the outcome a_i of every truthful mechanism satisfies $a_i(t_i, t_{-i}) \in \text{argmax}_a\{v_i(a_i) - p_i(a_i, t_{-i})\}$.*

Proof. Suppose towards a contradiction that there exists a type t such that for some allocation a'_i we have $v_i(a_i(t_i, t_{-i}), t_i) - p_i(a_i(t_i, t_{-i}), t_{-i}) < v_i(a'_i, t_i) - p_i(a'_i, t_{-i})$. If t'_i is such that $a_i(t'_i, t_{-i}) = a'_i$ then player i would have incentive to falsely declare t'_i. □

1.6 The VCG mechanism

Definition 3. The VCG mechanism is the one that implements the social choice function of selecting the outcome for which the sum of the valuations of all players

is maximum:

$$a(v) \in \operatorname*{argmax}_{a \in \mathcal{A}} \sum_{i=1}^{n} v_i(a).$$

The corresponding payment is

$$p_i(v_1, \ldots, v_n) = -\sum_{j \neq i} v_j(a(v)) + h_i(v_{-i})$$

so that each player pays an amount equal to the sum of the values of the other players. In this way the utility of each player becomes: $u_i(a_i) = v_i(a_i) - p(a_i, v_{-i}) = \sum_{i=1}^{n} v_i(a)$. That is the utilities of all players are equal to the social welfare.

Even though $h_i(v_{-i})$ is not of great significance if we impose some restriction on the payments then it takes a very particular form. For example in an auction it would be natural to demand the following restrictions on payments:

Definition 4. A mechanism satisfies *voluntary participation* if the utility of the players is always non-negative.

Definition 5. A mechanism has *no positive transfers* if no player is ever paid money.

Definition 6. The choice $h_i(v_{-i}) = \max_{b \in \mathcal{A}} v_i(b)$ is called the Clark pivot rule. This makes the payment of player i: $p_i(v) = \max_b \sum_{j \neq i} v_i(b) - \sum_{j \neq i} v_i(a(v))$.

The idea is that player i pays the total damage that he causes to the other players, i.e. the difference between the total welfare of the other players when he participates and when he doesn't participate. Of course there are many problems in mechanism design where players get payed in order to participate. A slight modification of the Clarke pivot rule also makes sense in most of these cases.

Lemma 2. *A VCG mechanism with Clarke Pivot payments makes no positive transfers. If $v_i(a) \geq 0$ for every $v_i \in V_i$ and $a \in \mathcal{A}$ then it also satisfies voluntary participation.*

Proof. Let a be the alternative maximizing $\sum_j v_j(a)$ and b the alternative maximizing $\sum_{j \neq i} v_j(b)$. The rule satisfies voluntary participation since

$$\begin{aligned} u_i(a) &= v_i(a) + \textstyle\sum_{j \neq i} v_j(a) - \sum_{j \neq i} v_j(b) \\ &\geq \textstyle\sum_j v_j(a) - \sum_j v_j(b) && (\text{as } v_i(b) \geq 0) \\ &\geq 0 && (\text{as } a \text{ maximizes } \sum_j v_j(a)). \end{aligned}$$

The Clarke pivot rule also satisfies no negative transfers since $p_i(v) = \sum_{j \neq i} v_i(b) - \sum_{j \neq i} v_i(a(v)) \geq 0$ and b maximizes this sum. □

Definition 7 (*Affine Maximizers or Generalized VCG*)**.** A generalization of this mechanism is the affine maximizer which weights with positive multipliers λ_i the values of each player and adds a constant γ_a to the value of each outcome a:

$$a(v) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{i=1}^{n} \lambda_i v_i(a) + \gamma_a.$$

These mechanisms are truthful for every domain. The payments (for the general domain) align the objective of each player l with the social choice function. This can be achieved when the payments are

$$p_l(v) = -\frac{1}{\lambda_l} \left(\sum_{i \neq l} \lambda_i v_i(a) + \gamma_a \right).$$

1.7 Implementation in Dominant Strategies

The mechanisms (direct revelation mechanisms) we have considered so far take as input the true values of the players. They achieve to elicit these secret values from the players by guaranteeing that no other strategy would be strictly more profitable than truth-telling. Perhaps there might be a different protocol, that takes as input lying strategies instead of true values and achieves a more desirable outcome? We will see now that if these lying strategies are dominant strategies, then there always exists a truthful mechanism with exactly the same outcome and payments. In other words, if we are interested in dominant equilibria, we can concentrate on truthful mechanisms without loss of generality.

We denote the lying strategies of player i as functions $s_i : T_i \to T_i$. A player with true type t_i can report any strategy $s_i(t_i)$ as his true type is known only to himself.

In dominant equilibria each player i with type t_i has a dominant strategy $s_i(t_i)$, that is, if all other players remain fixed to their declarations $s_{-i}(t_{-i})$ and player i changes his declaration to $s'_i(t_i)$ he cannot (strictly) increase his utility. Notice that a player prefers his dominant strategy even if the other players do not tell the truth.

Definition 8. A strategy $s_i(v_i) \in \mathbb{R}^m$ is called a *dominant strategy* for player i, if for every other strategy $s'_i(v_i) \in \mathbb{R}^m$ we have:

$$u_i(t_i, s_i(t_i), s_{-i}(t_{-i})) \geq u_i(t_i, s'_i(t_i), s_{-i}(t_{-i})).$$

But why are we interested in dominant strategies and not some other equilibrium concept? In any social choice problem the ideal would be to impose a solution that is optimal for each one of the players. As this is in most cases impossible we try to choose a solution that is an equilibrium so that even the players who are not totaly satisfied cannot impose a solution they would prefer. In game theory there are many equilibrium concepts. An equilibrium concept assumes the way players would react to a strategic situation. If the strategies chosen by the players are such that none of the possible, according to the specific equilibrium concept, reactions can improve any player's situation, then this set of strategies is an equilibrium point of the game. Dominant strategies is a simple and natural equilibrium concept however it is not used very often in Game Theory, because the existence of dominant strategies is a stringent condition. For many naturally-occurring games dominant strategies do not exist. Dominant strategies are not very interesting from the game-theoretic point of view because players are more predictable. However in mechanism design the goal is to start from an unpredictable game theoretic situation and and create a game where the players' actions will be predictable, and for this reason dominant strategies are the most preferred equilibrium concept.

1.8 The Revelation Principle

Theorem 2 (The Revelation Principle)**.** *For every mechanism in dominant strategies there exists a truthful mechanism, such that for every possible input the two mechanisms produce the same allocation and payments.*

Proof. The idea of proof is that given a mechanism in dominant strategies we construct a truthful one as follows: the truthful mechanism takes as input the true values of the players and simulates their lying strategies. So if $s_1(t_1), \ldots, s_n(t_n)$ are the dominant strategies of the mechanism $(a, p_1, \ldots, p_n)$ we construct a new social choice function $f(t_1, \ldots, t_n) := a(s_1(t_1), \ldots, s_n(t_n))$ (that is $f := a \circ s$) and similarly $p'_i(t_1, \ldots, t_n) = p_i(s_1(t_1), \ldots, s_n(t_n))$. □

1.9 Outline of the Thesis

In the first chapter we gave an introduction to Mechanism design. From the second chapter and until the end of the Thesis we deal specifically with the scheduling selfish unrelated machines problem as all results in this thesis concern this specific problem. In the second chapter we state the problem and start presenting some relevant fundamental facts and results. We consider a number of different approaches to it. On the way we also present some of the tools we have developed for studying the problem [19, 31]. Then we present all known algorithms for the problem and show some easy lower bounds namely a 2-lower bound for the case of two machines and an n lower bound for some special classes of algorithms. Two of these classes are the class of additive and the class of threshold mechanisms. We show that the class of threshold mechanisms is identical to the class of additive mechanisms (we announced the result in [18] and here we provide the complete proof).

In the third chapter we explore the possible truthful mechanisms for the scheduling problem. We develop some tools for the general case of m tasks but we only manage to characterize the possible mechanisms for the case of three tasks. We believe that this work provides a better intuition about truthful mechanisms. As the existing lower bound for the case of 3 players uses the result for the two-task case, this work is potentially useful not only for characterizing truthful mechanisms for more than two players, but also for obtaining new lower bounds. This chapter is based on an unpublished working paper.

In the fourth chapter we give a proof a lower bound $1 + \sqrt{2} \approx 2.41$ on the approximation ratio of truthful mechanisms for the case of three or more machines. This chapter is based primarily on the paper "A lower bound for scheduling mechanisms" [19] co-authored with George Christodoulou and Elias Koutsoupias. (The same result can also be found in [15, 17] however the proof presented here is different and shorter.)

In the fifth chapter we give a proof of a $1+\phi \approx 2.618$ lower bound as the number of machines n tends to ∞. Our technique also gives improved lower bounds for any constant number of machines $n \geq 4$. This chapter is based primarily on the paper "A lower bound of $1 + \phi$ for truthful scheduling" [31] co-authored with Elias Koutsoupias.

The objective in the scheduling problem is to minimize the makespan, i.e. to

minimize the maximum completion time, or in other words to minimize the L_∞ norm of the machine loads. On the other hand the goal achieved by the VCG, which is the best known algorithm, is that of minimizing the sum of completion times, in other words to minimize the L_1 norm of the machine loads. It turns out that our techniques can be easily adapted in order to get lower bounds for all L_p norms $2 \leq p < \infty$.

Finally in the last chapter we provide a characterization for the case of two-players and arbitrarily many tasks. We show that the class of truthful mechanisms is very limited: A decisive truthful mechanism partitions the tasks into groups so that the tasks in each group are allocated independently of the other groups. Tasks in a group of size at least two are allocated by an affine minimizer and tasks in singleton groups by a threshold mechanism (which is however a task-independent mechanism except for countably many points). This characterization is about all truthful mechanisms, including those with unbounded approximation ratio. A direct consequence of this approach is that the approximation ratio of mechanisms for two players, for the objective of minimizing the makepan, is 2, even for two tasks. In fact, it follows that for two players, VCG is the unique algorithm with optimal approximation 2. This characterization provides some support that any decisive truthful mechanism (for 3 or more players) partitions the tasks into groups some of which are allocated by affine minimizers, while the rest are allocated by a threshold mechanism (in which a task is allocated to a player when it is below a threshold value which depends only on the values of the other players). This chapter is based primarily on the paper "A characterization of 2-player mechanisms for scheduling" [18] co-authored with with George Christodoulou and Elias Koutsoupias.

Chapter 2

An introduction to the Scheduling Problem

> "No resource allocation mechanism can ensure a fully efficient level of public goods, because it is in the selfish interest of each person to give false signals, to pretend to have less interest in a given collective activity than he really has."
>
> Paul Samuelson

2.1 Statement of the problem

The scheduling problem on unrelated machines is one of the most fundamental scheduling problems [27, 28].

Nisan and Ronen introduced the mechanism-design version of the problem in their paper that founded the algorithmic theory of Mechanism Design [42, 43]. They gave a truthful n-approximate (polynomial-time) algorithm and conjectured that there is no deterministic mechanism with approximation ratio less than n. They also showed that no mechanism (polynomial-time or not) can achieve approximation ratio better than 2.

In the scheduling problem you can imagine a very big company (the mechanism designer) that needs very badly to process a set of m tasks as soon as possible (all tasks have to be allocated)–no matter what the cost will be– using n machines.

The company pays the machines for the tasks they process. Each task has to be processed by exactly one machine, but different tasks can be processed in parallel on different machines.

Definition 9 (The scheduling unrelated machines problem)**.** The input to the scheduling problem is a nonnegative matrix t of n rows, one for each machine-player, and m columns, one for each task. The entry t_{ij} (of the i-th row and j-th column) is the time it takes for machine i to execute task j. Let t_i denote the times for machine i, which is the vector of the i-th row. The output is an allocation $a = a(t)$, which partitions the tasks into the n machines. We describe the partition using indicator values $a_{ij} \in \{0, 1\}$: $a_{ij} = 1$ iff task j is allocated to machine i. We should allocate each task to exactly one machine, or more formally $\sum_{j=1}^{m} a_{ij} = 1$. The goal is to minimize the makespan, i.e. to minimize the total processing time of the player that finishes last.

Definition 10. We will say that an algorithm ALG achieves an r-approximation if for any possible input t we have $\frac{cost(ALG(t))}{OPT(t)} \leq r$, where $OPT(t)$ is the cost of the optimal allocation for input t.

In the mechanism-design version of the problem we consider direct-revelation mechanisms. That is, we consider mechanisms that work according to the following protocol:

- Each player i declares the values in row t_i, which is known only to player i.
- The mechanism, based on the declared values, decides how to allocate the tasks to the players.
- The mechanism, based on the declared values, and the allocation of the previous step, decides how much to pay each player.

The mechanism consists of two algorithms, an allocation algorithm and a payment algorithm. The cost of a player (machine) is the sum of the times of the tasks allocated to it minus the payment. One way to think of it is as if the players are lazy and don't want to execute tasks, and the mechanism pays them enough to induce them to execute the tasks. On the other hand, the players know both the allocation and the payment algorithm and may have an incentive to lie in the first step. We are

interested in mechanisms in which there exists a dominant strategy for each one of the players. Thanks to the Revelation Principle (see Section 1.7) every mechanism in dominant strategies can be turned into an equivalent truthful mechanism. This allows us to concentrate only on truthful mechanisms.

At a first reading the mechanism design version of the problem might seem a bit unnatural because of the payments given to the machines in order to make them tell their true types. Payments in auctions seem perhaps better justified and natural compared to the payments in scheduling, but in fact in the auction settings studied in traditional mechanism design, payments are not given in order to maximize the revenue of the auctioneer, as one would perhaps first guess. The goal of the auctioneer is not to gain as much as possible but to maximize the utility of the players. Payments are in fact nothing else than a way to force the players to tell the truth. The purpose of the payments in scheduling is along the lines of purpose of the payments in auctions. The mechanism designer pays the players so that they tell their true processing times. So in the scheduling problem the players get paid in order to do a set of jobs (without payments the machines would lie and say they cannot perform any job). In fact, the scheduling problem is in many aspects like the multi-unit combinatorial auction problem.

2.2 An n-approximate truthful mechanism

For someone who is familiar with the VCG mechanism for multi-unit auctions it is easy to observe that the VCG mechanism we devised for auctions can be applied to scheduling giving an n-approximation of the goal of minimizing the makespan.

Definition 11. The MinWork mechanism [42]

- *Allocation:* Consider each task separately and allocate it to the the player with the minimum processing time.
- *Payment:* The payment a player receives for each task allocated to him is equal to the processing time of the second best player for this task.

The MinWork Mechanism coincides with the VCG mechanism, because it achieves the goal of selecting the outcome that minimizes the sum of processing times of all players:

Definition 12 (The VCG mechanism[47, 20, 25])**.** The VCG mechanism for the scheduling problem chooses an allocation a such that $a \in \text{argmin}_a\{\sum_{i=1}^{n} a_i \cdot t_i\}$.

But how well does a mechanism that minimizes the sum of processing times perform for the objective of minimizing the makespan?

Example 5. A worst-case example for the VCG mechanism is the following $n \times n$ matrix where the allocation produced by the VCG mechanism is indicated by the stars and ϵ is some very small positive value:

$$\begin{pmatrix} 1-\epsilon\star & 1-\epsilon\star & \dots & 1-\epsilon\star \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}.$$

The optimal allocation would be a diagonal allocation with cost 1 allocating one task to each player and the resulting approximation ratio is n.

Theorem 3 ([42])**.** *The MinWork mechanism achieves an n-approximation.*

Proof. The cost of the optimal allocation OPT satisfies $\text{OPT} \geq \frac{1}{n}\sum_{j=1}^{k} \min_i t_{ij}$. And as the makespan of MinWork is no more than $\sum_{j=1}^{k} \min_i t_{ij}$ (in the case when one of the players is the fastest for all tasks, like in the previous example) we get that the approximation ratio of MinWork is n. □

Remark 2. The same idea gives a lower bound of at least n for all affine minimizers, namely if the affine minimizer has weights $\lambda_1, \dots, \lambda_n$ we take the input matrix

$$\begin{pmatrix} \frac{b}{\lambda_1}-\epsilon\star & \frac{b}{\lambda_1}-\epsilon\star & \dots & \frac{b}{\lambda_1}-\epsilon\star \\ \frac{b}{\lambda_2} & \frac{b}{\lambda_2} & \dots & \frac{b}{\lambda_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{b}{\lambda_n} & \frac{b}{\lambda_n} & \dots & \frac{b}{\lambda_n} \end{pmatrix}$$

for some sufficiently large $b >> \max_{a\in\mathcal{A}} \gamma_a$.

In fact:

Theorem 4. *An affine minimizer that has at least two different allocations a, a' with $\gamma_a \neq \gamma_{a'}$ has unbounded approximation ratio.*

Proof. To see this, fix an affine minimizer for which there exist $\gamma_a < \gamma_{a'}$ and consider the following instance:

$$t_{ij} = \begin{cases} b & \text{if } a_{ij} = 1 \\ \epsilon & \text{if } a_{ij} = 0, \end{cases}$$

where $b < \dfrac{\gamma_{a'} - \gamma_a}{n}$. The affine minimizer achieves approximation ratio at of least b, (as $\sum_{i=1}^{n} a_i t_i + \gamma_a \leq n \cdot b + \gamma_a < \epsilon + \gamma_{a'} = \sum_{i=1}^{n} a'_i t_i + \gamma_{a'}$) while the optimum would have been exactly ϵ. □

2.3 Monotonicity: A local characterization of Truthfulness

We discuss below the Monotonicity Property, a simple necessary and sufficient condition for truthfulness. This is true for every finite convex domain, but we restrict the discussion to the scheduling domain.

Definition 13 (Monotonicity Property)**.** An allocation algorithm is called monotone if it satisfies the following property: for every two sets of tasks t and t' which differ only on machine i (i.e., on the i-the row) the associated allocations a and a' satisfy

$$(a_i - a'_i) \cdot (t_i - t'_i) \leq 0,$$

where $\cdot$ denotes the dot product of the vectors, that is, $\sum_{j=1}^{m} (a_{ij} - a'_{ij})(t_{ij} - t'_{ij}) \leq 0$.

The Monotonicity Property states that when for fixed values of the other players we increase the times of the tasks for player i, his allocation can only become smaller, i.e. the allocation function of player i is decreasing with respect to the processing times of player i. Notice that the Monotonicity Property involves only the allocation of one player (the i-th player).

Theorem 5 (Saks and Yu)**.** *A mechanism is truthful if and only if its allocations satisfy the Monotonicity Property.*

The short proof that follows shows that the Monotonicity Property is necessary condition for truthful mechanisms. Saks and Yu [46] gave a proof for the other (considerably more difficult) direction, that it is also a sufficient condition. In fact,

they showed a much more general result: the property is sufficient for every finite convex domain; this includes the unrestricted domain and the combinatorial auction domains.

Proof. (if direction) First remember that from Lemma 1 the payments cannot depend directly on the declaration t_i of player i, but only indirectly through his allocation $a_i(t)$ and the declarations t_{-i} of the other players, that is, $p_i(t) = p_i(a_i(t), t_{-i})$.

When player i has valuations t_i, he has no incentive to declare t'_i when

$$t_i \cdot a_i - p_i(a_i, t_{-i}) \leq t_i \cdot a'_i - p_i(a'_i, t_{-i})$$

Similarly, when we inverse the roles of t and t', we have

$$t'_i \cdot a'_i - p_i(a'_i, t_{-i}) \leq t'_i \cdot a_i - p_i(a_i, t_{-i})$$

Now if we add the above inequalities, we get the Monotonicity Property. □

The implications are that we don't have to consider at all the payment algorithm. If an allocation function satisfies the Monotonicity Property then we are sure that there exist payments that together with the allocation function make a truthful mechanism. To prove lower bounds or to design good mechanisms, we can completely forget about mechanisms, payments, truthfulness etc, and simply focus on the subclass of monotone allocation algorithms. The difference from traditional algorithm design is that we have to consider the whole space of inputs together. This is because the property should be satisfied by any two pairs of input with the corresponding output.

Definition 14. We will denote by R_a^i the **closure** of the subset of $\mathbb{R}^m$ where the mechanism gives assignment a to machine i for some fixed t_{-i} and we will call it a *region of the mechanism.*

For the scheduling domain the monotonicity property has a straightforward geometric interpretation. Fix the values of all players except for player i. For warmup suppose first that there is only one task and consider how the allocation of player i changes as his processing times become bigger. For small values of t_i player i gets the task, then there is a threshold and his allocation changes once and for all from one to zero. This threshold might depend on the values of the other players t_{-i}, but

it does not depend on the values of player i. The reason is that all inputs t_i with allocation 1 and inputs t'_i has allocation 0 according to the Monotonicity Property should satisfy $t_i \leq t'_i$.

Definition 15. Let $f^i_{a:a'}(t_{-i}) := \sup\{(a' - a) \cdot t'_i \mid t'_i \in R^i_{a'}\}$.

Note that for fixed t_{-i}, $f_{a':a}(t_{-i})$ is constant.

Lemma 3. *Regions R_a and $R_{a'}$ are separated by the hyperplane $(a'_i - a_i) \cdot t_i = f^i_{a:a'}(t_{-i})$ and each region is bounded by a convex polytope.*

Proof. By the Monotonicity Property, for every $t_i \in R_a$ and $t'_i \in R_{a'}$ we must have $(a - a') \cdot (t_i - t'_i) \leq 0$. Equivalently we can write $(a' - a) \cdot t_i \geq (a' - a) \cdot t'_i$. Since $f^i_{a:a'}(t_{-i}) = \sup\{(a' - a) \cdot t'_i \mid t'_i \in R^i_{a'}\}$ we have $(a' - a) \cdot t_i \geq f^i_{a:a'} \geq (a' - a) \cdot t'_i$ and the lemma follows. □

Now let us consider the case of two tasks, again for fixed values of the other players t_{-i}. We consider every possible (t_{i1}, t_{i2}), that is we consider the space of all possible valuations for a player i. That is the two axes are t_{i1} and t_{i2} and the mechanism partitions the plane into a different region, for each one of the four possible allocations of player i. In particular, let $R_{a_{i1}a_{i2}}$ denote the set of inputs of player i for which the mechanism has allocation (a_{i1}, a_{i2}) for the i-th player.

If t_i, t'_i are two inputs and a_i, a'_i their corresponding allocations the Monotonicity Property gives that $(a_i - a'_i) \cdot t_i \leq (a_i - a'_i) \cdot t'_i$ as we saw in Lemma 3 this is equivalent to saying that the boundary between $R_{a_{i1}a_{i2}}$ and $R_{a'_{i1}a'_{i2}}$ is of the form $(a'_{i1} - a_{i1})t_{i1} + (a'_{i2} - a_{i2})t_{i2} = f_{a':a}(t_{-i})$ where $f_{a:a'}(t_{-i})$ is a constant for fixed t_{-i}. For example the boundary between regions R_{11} and R_{00} is of the form $t_{i1} + t_{i2} = f_{00:11}(t_{-i})$. Since the allocation variables a_{i1} and a_{i2} are either 0 or 1, the boundaries have very specific slopes. Therefore the allocation of the mechanism should have one the 2 forms of Figure 4.1.

To get the whole picture every mechanisms creates a partition of $\mathbb{R}^{n \times m}$ to n^m regions, one for each possible allocation of the tasks to the players. Monotonicity is a necessary and sufficient condition for Truthfulness, and the geometrical interpretation of monotonicity is that any m-dimensional cut of the mechanism, for constant t_{-i}, has boundaries of the specific form given by the monotonicity property.

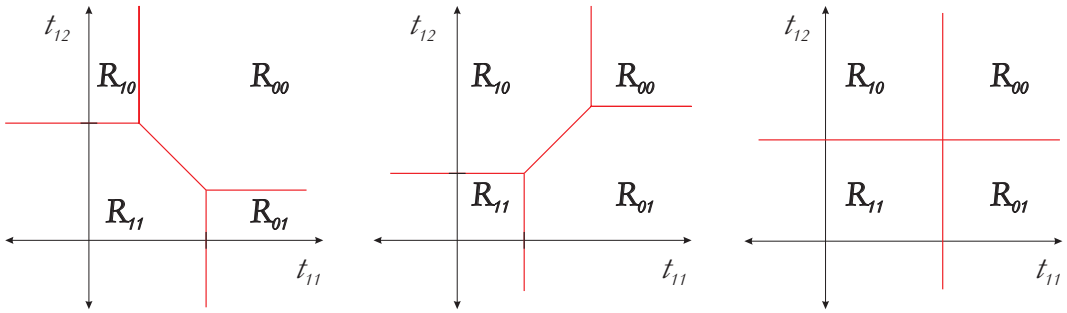

Figure 2.1: The two possible ways to partition the positive orthant and the threshold mechanism as a degenerate case of both.

Notice that the Monotonicity Property does not specify what happens exactly at the boundaries between regions.

In fact, in order to obtain lower bounds [17, 31] we use repeatedly a restricted but easy to use variant of the monotonicity property, which gives a way to change the values of one player in order to keep his allocation fixed. There is a very simple and nice geometrical interpretation of the following lemma: If we know which is the assignment of player i for some input values t then we know the assignment of a whole box (see Figure 2.2 for a 2-dimensional example) on the plane for constant t_{-i}. Which is this box depends on the assignment a of the point.

Lemma 4 ([17]). *Let t be a matrix of processing times and let $a = a(t)$ be the allocation produced by a truthful mechanism.*

a. *Suppose that we change only the processing times of machine i and in such a way that $t'_{ij} > t_{ij}$ when $a_{ij} = 0$, and $t'_{ij} < t_{ij}$ when $a_{ij} = 1$. The mechanism does not change the allocation to machine i, i.e., $a_i(t') = a_i(t)$. (However, it may change the allocation of other machines).*

b. *Fix now a mechanism with approximation ratio r and consider an instance whose optimal allocation has cost c. Suppose that for a task j we have $t_{ij} = 0$ for machine i, and $t_{i'j} = \infty$ for every other machine i', where ∞ denotes a very large real number, greater than $r \cdot (c + u)$, for some constant u. If we change the values of machine i for all other tasks as in the first part of the lemma but raise the time for task j to $t'_{ij} = u$, the mechanism again does not change the allocation vector of machine i.*

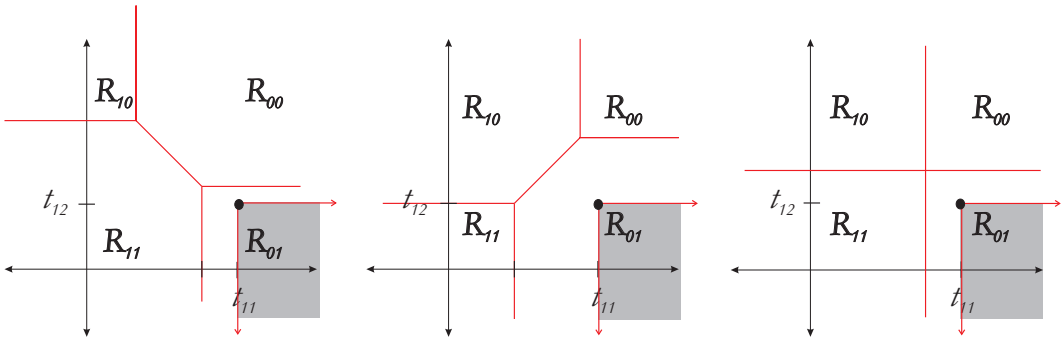

Figure 2.2: A geometrical interpretation of Lemma 4: If we know that the assignment of the black dot is 01 then we can deduce that the allocation in the interior of the gray region is also the same, no matter what is the particular shape of the mechanism (we increase the value of the task with assignment 0 and decrease the value of the task with assignment 1). If the black point is not on a boundary then we can additionally say that the assignment in the closure of the box is also the same. This remark will play a very significant role in our proof of the 2-player characterization.

Proof. a. By the Monotonicity Property, we have

$$\sum_{j=1}^{m}(t_{ij} - t'_{ij})(a_{ij}(t) - a_{ij}(t')) \leq 0.$$

For the first property, observe that all terms of the sum are nonnegative (by the premises of the lemma). The only way to satisfy the inequality is to have all terms equal to 0, that is, $a_{ij}(t) = a_{ij}(t')$.

b. When we change the value t_{ij} to u, the optimum makespan becomes at most $c + u$. If the mechanism allocates task j to a machine different than i, the approximation ratio is greater than r, which contradicts the hypothesis about the mechanism. Therefore $a_{ij}(t) = a_{ij}(t') = 1$, which makes the term corresponding to task j in the sum $\sum_{j=1}^{m}(t_{ij} - t'_{ij})(a_{ij}(t) - a_{ij}(t')) \leq 0$ vanish. For the rest of the terms we repeat the argument of the first part.

□

To be more formal the geometrical equivalent of the preceding lemma can be stated as follows:

Definition 16. The *Minkowski sum* of two sets $A, B \subseteq \mathbb{R}^n$ is $A \oplus B = \{a + b \mid a \in A, b \in B\}$.

Definition 17. Let $B_a^i := \{t_i \mid (-1)^{a_j} t_{ij} \geq 0,\ j = 1, \ldots, m\}$.

For $m = 2$ each B_a^i is a quadrant of $\mathbb{R}^2$.

Lemma 5. *Take two different assignments a, a'. If a point b belongs to region R_a^i of a truthful mechanism, then also $b \oplus B_a \subseteq R_a$.*

A useful observation that was first used in [17] is the following:

Remark 3. Suppose a mechanism for m tasks is truthful for player i. If we keep the processing times of player i for a set of k tasks constant then the mechanism for the rest of the tasks is a truthful mechanism for $k - m$ tasks.

The reason is that if we fix the values of some tasks, some of the terms in the Monotonicity Property sum become zero and Monotonicity holds for the rest of the tasks so that we still get a truthful mechanism for the rest of the tasks.

In the context of combinatorial auctions, [34] proposed a stronger condition very similar to the Monotonicity Property. This condition however is not satisfied by all truthful mechanisms.

Definition 18 (Strict Monotonicity (S-Mon))**.** An allocation algorithm is called strictly monotone if it satisfies the following property: for every two sets of tasks t and t' which differ only on machine i (i.e., on the i-the row) such that the associated allocations a and a' satisfy $a \neq a'$ we have

$$(a_i - a'_i) \cdot (t_i - t'_i) < 0,$$

where $\cdot$ denotes the dot product of the vectors, that is, $\sum_{j=1}^{m} (a_{ij} - a'_{ij})(t_{ij} - t'_{ij}) < 0$.

For the scheduling domain there exist truthful mechanisms, even with $O(n)$ approximation ratio (n is the upper bound on the approximation ratio) that do not satisfy this condition (see Example 6 for a non S-Mon mechanism). There is a strong connection between this property and Arrow's "Independence of Irrelevant Alternatives" condition [29] in the context of voting systems. To be more specific the Monotonicity Property together with IIA imply Strong Monotonicity.

Definition 19 (Independence of Irrelevant Alternatives (IIA))**.** A social choice function satisfies IIA if for any two types t, t' such that the associated allocations a and a' satisfy $a \neq a'$ then there exists a player i with

$$(a_i - a'_i) \cdot (t_i - t'_i) \neq 0.$$

For example, if the processing times t_i of a single machine change to t'_i, then either the allocation a_i changes to some allocation a'_i with $a'_i \neq a_i$ or the whole allocation remains the same, i.e. $a' = a$.

Arrow himself writes that "the austerity imposed by this condition is perhaps stricter than necessary". However one might hope that imposing this additional condition would perhaps at least give us some insight for how to deal with the general case. As we will see in Section 2.6 it is very easy to prove an n-lower bound for the class of S-Mon mechanisms. Still a characterization of all S-Mon scheduling mechanisms for more than two players is not known yet and would be an interesting result.

2.4 Known mechanisms for the Scheduling Problem

There are two ways to search for truthful mechanisms: the first is to concentrate on the social choice function: For which social choice functions are there payment functions so that the resulting mechanism is truthful? The other one is to concentrate on the payment functions: Assuming payments that satisfy a certain condition, how well can the social goal be approximated?

Starting from the allocation part

In light of the fact that the Monotonicity Property is a necessary and sufficient condition for truthfulness (see Theorem 5), which does not refer to payments at all, when trying to construct a mechanism we usually only care about constructing an allocation function that satisfies Monotonicity. For this reason special mechanism classes are usually described using their allocation part.

The mechanisms that are known to be truthful can be put in two classes: affine minimizers and task-independent mechanisms. These two classes demonstrate two

different ways in which we can generalize the VCG mechanism preserving truthfulness.

A first idea to generalize the VCG is the following: The VCG achieves the goal of maximizing the sum of the valuations of the players. As we have already seen in Definition 7 if we apply a linear transformation to the valuation of each one of the players we get the affine minimizer or generalized VCG:

Affine Minimizer (or Generalized VCG). An affine minimizer generalizes the VCG by weighting with positive multipliers λ_i the values of each player and adding a constant γ_a to the value of each outcome a:

$$a(t) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{i=1}^{n} \lambda_i a_i \cdot t_i + \gamma_a.$$

The payments are

$$p_l(a, t_{-l}) = -\frac{1}{\lambda_l} \left(\sum_{i \neq l} \lambda_i a_{ij} t_{ij} + \gamma_j \right).$$

If all constants γ_a are zero then the term weighted VCG is often used. The constants γ_a change the picture of the mechanism dramatically: The VCG mechanism has no sloped lines, while the affine minimizer has a sloped line in one of its projections constant t_{-i} see Figure 4.1 (provided that the γ_as are not all equal to each other).

A second way to generalize the VCG is by maintaining one of its basic properties (better understood if you see the MinWork description), namely that it allocates each task independently of the rest.

Definition 20 (Task-independent mechanism)**.** Another interesting class of mechanisms for the scheduling problem are the task independent mechanisms: Each task is allocated independently of the remaining tasks.

Note that there exist mechanism that allocate each task independently but are not truthful because they do not satisfy the monotonicity property.

Task-independent mechanisms are special cases of threshold mechanisms. The idea behind threshold mechanisms is that for each pair of tasks the picture of the mechanism (for constant values of the other players and tasks) would be the like in the 3rd case of Figure 4.1, that is the picture would have no sloped lines. As we will see just the absence of these sloped lines is enough to prove an n lower bound for the objective of minimizing the makespan.

Definition 21 (Threshold mechanism)**.** A threshold mechanism for the scheduling domain is one for which there are threshold functions h_{ij} such that the mechanism allocates item j to player i if and only if $t_{ij} \geq h_{ij}(t_{-i})$. What distinguishes these mechanisms from general mechanisms is that the thresholds depend only on the values of the other players but not on the other values of the player himself. In threshold mechanisms there is a single threshold for getting or not task j and it is the same regardless of the rest of the tasks allocated to player i. It is not true in general that every set of functions h_{ij} defines a legal mechanism, as they have to be consistent between them. In particular, the threshold functions should be such that every item j is allocated to exactly one player. In other words, exactly one of the constraints $v_i(\{j\}) \geq h_{ij}(v_{-i})$, for $i = 1, \ldots, n$, should be satisfied.

Example 6 (Threshold but not Task-independent). The following mechanism for 3 players and 2 tasks is a threshold mechanism but not a task-independent mechanism:

Give the first task to the player with the minimum value and process it in time $\min\{t_{11}, t_{21}, t_{31}\}$. Give the second task to player 1 if $t_{12} \leq \min\{t_{22}, t_{32}\}$, otherwise if $t_{22} < t_{32}t_{11}$ give the second task to the player 2, else give it to player 3.

Note here that the values of the processing times, but not the allocation, of player 1 for the first task affect the allocation of the second task.

Starting from the payments

Nissan and Ronen in [42] define a class of mechanisms called "additive mechanisms" for which they could provide an n lower bound for any approximation algorithm in terms of conditions satisfied by their payment functions. A natural question that arises is to describe the allocation rule of these types of mechanisms without any reference to their payment rule. Theorem 6 gives such a description.

Definition 22 ([42])**.** A mechanism is called *additive* if for each agent i, processing times matrix t and allocation $a = a(t)$ of tasks the payment of agent i is

$$p_i(a_i, t_{-i}) = \sum_{j=\{1,\ldots,m\}} a_{ij} p_i(e_j, t_{-i}).$$

As you can see in an additive mechanism a player is payed for each task he processes separately, that is his payment for one task does not depend on what other tasks are assigned to him. For example $p_i(11, t_{-i}) = p_i(10, t_{-i}) + p_i(01, t_{-i})$

This assumption does not allow for a mechanism that gives a bonus payment to a player who is useful and processes many jobs, neither for a mechanism that demands a discount from a player that gets a lot of jobs and subsequently a big payment.

2.5 Additive and threshold mechanisms

The definition we gave for threshold mechanisms involves only the allocation and the definition of additive mechanisms involves only the payments. We show that the class of additive mechanisms coincides with the class of threshold mechanisms.

Theorem 6. *A mechanism is additive iff it is a threshold mechanism.*

Proof. Fix a player i and the values t_{-i} of the other players, for simplicity we will write $p(a)$ instead of $p_i(a_i, t_{-i})$. Take two allocations a and a', that differ only on task k (i.e. $a'_k = 1 - a_k$), then

$$p(a') - p(a) = \Big[(1-a_k)p(e_k) + \sum_{\substack{j=1 \\ j\neq k}}^{m} a_j p(e_j)\Big] - \sum_{j=1}^{m} a_j p(e_j) \quad \text{(the mechanism is additive)}$$
$$= (1-2a_k)p(e_k) = (-1)^{a_k} p(e_k)$$

Since the mechanism is truthful, if the value of player i is $t_i \in R_a$, it should be $p(a) - at_i \geq p(a') - a't_i$. Taking two allocations a and a', that differ only on task k, and rearranging the previous inequality we get $t_{ik} \leq (-1)^{a_k+1} p(e_k)$ for every $t_i \in R_a$ and every $k = 1, \ldots, m$. Let

$$F_a := \{t_i \mid t_{i1} \leq (-1)^{a_1+1} p_i(e_1), \ldots, t_{im} \leq (-1)^{a_m+1} p_i(e_m)\}$$

These sets satisfy $R_a \subseteq F_a$ and $F_a \cap B_b = \emptyset$, for any two allocations a, b with $a \neq b$. Finally as the mechanism is a partition of the input space, we get that $R_a = F_a$. It is now easy to see that the mechanism is a threshold mechanism: player i gets task k if and only if $t_{ik} \leq p_i(e_k, t_{-i})$. □

2.6 Some easy lower bounds

As we will see it is easy to show a lower bound of 2 for two (or more) machines and three (or more) tasks.

Theorem 7 ([42]). *There does not exist a mechanism that achieves an r-approximation of the scheduling problem with three or more tasks for any $r < 2$.*

On what follows we put a $\star$ next to the value of the player who gets each one of the tasks.

Proof. We start with the instance $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$, which basically admits two allocations (because of its symmetry):

- Either the allocation splits the tasks between the players and by the Monotonicity Property and especially by applying Lemma 4 we get that

$$\begin{pmatrix} 1\star & 1\star & 1 \\ 1 & 1 & 1\star \end{pmatrix} \rightarrow \begin{pmatrix} 1\star & 1\star & 1 \\ 1 & 1 & 0\star \end{pmatrix},$$

- or one player gets all tasks and again by Lemma 4

$$\begin{pmatrix} 1\star & 1\star & 1\star \\ 1 & 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1\star & 1\star & 0\star \\ 1 & 1 & 1 \end{pmatrix}.$$

In any case the cost of the resulting allocation is 2, while the cost of the optimal allocation would have been 1, which gives a lower bound of 2.

Remark 4. In fact when we lower the processing time of a player for a task he gets, we also make the values of the tasks with assignment 0 a little bigger so that Lemma 4 holds.

□

Theorem 8. *Every threshold mechanism for at least $n^2 - n + 1$ tasks has approximation ratio at least n.*

As we have seen in Theorem 6 additive and threshold mechanisms coincide so we also get the following theorem:

Theorem 9 ([42]). *Every additive mechanism for at least $n^2 - n + 1$ tasks has approximation ratio at least n.*

Proof. For a better understanding we will give the proof of a lower bound of 3 for the case of 3 tasks. Exactly the same technique gives a lower bound of n for $n^2 - n + 1$ tasks and n players as this number of tasks guarantees that one of the players will get at least n tasks.

We start with the instance

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

where we can assume w.l.o.g. that player 1 gets at least 3 tasks. The idea is to lower all values of player 1 to some small $\epsilon > 0$ except for 3 (and in the general case n) values. If we set a task that gets allocated to player 1 to ϵ, then by Lemma 4 the allocation of player 1 remains the same. We then lower one by one all of the tasks that are not assigned to player 1. Player 1 doesn't loose any of the tasks initially assigned to him, because in a threshold mechanism we cannot move from one region R_a to another region $R_{a'}$ that differs from the original region in more than one position by lowering a single task (This might have not been true if the projections of the mechanism had some sloped lines). We continue until all tasks of player 1 except for the 3 that were initially assigned to him are zero. The approximation ratio is 3.

Example 7. For example if the original assignment is the one marked by the stars then using Lemma 4 we get:

$$\begin{pmatrix} 1\star & 1\star & 1\star & 1 & 1 & 1 & 1\star \\ 1 & 1 & 1 & 1\star & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1\star & 1\star & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1\star & 1\star & 1\star & 0 & 0 & 0 & 0\star \\ 1 & 1 & 1 & 1\star & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1\star & 1\star & 1 \end{pmatrix}$$

Exactly the same technique gives a lower bound of n for $n^2 - n + 1$ tasks and n players as this number of tasks guarantees that one of the players will get at least n tasks. □

Theorem 10 ([42, 40])**.** *Every Strictly Monotone mechanism for at least $n^2 - n + 1$ tasks has approximation ratio at least n.*

Proof. The proof just extends the idea of the lower bound of 2 proof (Theorem 7) for more than two players. The reason that this proof does not work for general mechanisms (satisfying Monotonicity but not S-Mon) is that in general if we change the

allocation of one player the rest of the players might exchange their tasks. (These players change allocation and satisfy the Monotonicity property with equality, because they have $t = t'$.) If we however assume S-Mon we are sure that the rest of the players will not exchange their tasks.

For example for 3 players we start again with the instance

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Assume that the allocation is indicated by the stars (the proof in the other cases is very similar). One player gets at least three tasks. The idea is to keep the values of these tasks intact and lower to zero the value of the player who gets each one of the rest of the tasks. We can do this without affecting the allocation by changing the values of one player at a time.

$$\begin{pmatrix} 1\star & 1\star & 1\star & 1 & 1 & 1 & 1\star \\ 1 & 1 & 1 & 1\star & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1\star & 1\star & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1\star & 1\star & 1\star & 1 & 1 & 1 & 0\star \\ 1 & 1 & 1 & 1\star & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1\star & 1\star & 1 \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{pmatrix} 1\star & 1\star & 1\star & 1 & 1 & 1 & 0\star \\ 1 & 1 & 1 & 0\star & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1\star & 1\star & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1\star & 1\star & 1\star & 1 & 1 & 1 & 0\star \\ 1 & 1 & 1 & 0\star & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0\star & 0\star & 1 \end{pmatrix}.$$

□

As an endnote observe that there is great similarity between these two proofs: both start with the same instance and exploit its symmetry. The basic difference is that in threshold mechanism it is enough to consider just one player, while in S-Mon mechanisms you have to consider and change the values of all players.

Definition 23. A mechanism has super-additive payments for player i if player i never gets paid less than the sum of the payments he would get for each of the tasks separately. This is equivalent to saying the region where player i gets all tasks is an m-dimensional box, i.e. it is of the form $R^i_{1\ldots1} = \{t_i | t_{ij} \leq h_{ij} \text{ for } j = 1 \ldots m\}$ where the constants h_{ij} may only depend on t_{-i}.

Theorem 11. *Every mechanism with super-additive payments has approximation ratio at least* $1 + \sqrt{n}$.

Proof. We will just show that it has ratio at least $\sqrt{n}$. By adding some additional dummy tasks (like we do in the proof of the $1+\sqrt{2}$ lower bound) it is an easy exercise to improve this to a lower bound of $1+\sqrt{n}$.

For $0 < b < 1$ consider the $n \times n$ matrix of processing times

$$\begin{pmatrix} b & \dots & b \\ 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{pmatrix}.$$

If the first player gets all tasks then the approximation ratio is $n \cdot b$. Consequently in order to achieve a ratio smaller than $n \cdot b$ we must have $h_{11} = f_{01\dots1:1\dots1} \leq b$. Similarly we have $h_{1j} = f_{1\dots101\dots1:1\dots1} \leq b$. Now if player 1 gets all tasks except for task j we raise the value of task j to $b+\epsilon$ and lower all other values of player 1 to ϵ:

$$\begin{pmatrix} \epsilon\star & \dots & b+\epsilon & \dots & \epsilon\star \\ 1 & \dots & 1\star & \dots & 1 \\ \vdots & \dots & 1 & \ddots & \vdots \\ 1 & \dots & 1 & \dots & 1 \end{pmatrix}.$$

By Lemma 4 the allocation of player 1 remains the same and the approximation ratio is $\frac{1}{b}$. Consequently in order to achieve ratio less than $\frac{1}{b}$ we must have $h_{ij} = f_{1\dots1:1\dots101\dots1} \geq b$. The solution of the equation $n \cdot b = \frac{1}{b}$ is $\sqrt{n}$. □

2.7 The allocation graph of each player

The allocation part of the mechanism gives a partition of $R_+^{n\times m}$ to n^m different regions, one for each possible different allocation a of the tasks to the players (remember that each task is allocated to exactly one player). Now the truthfulness requirement imposes restrictions on the geometry of the projections of this mechanism for fixed t_{-i}, because it describes the game-theoretic situation player i faces: Should he deviate and lie when all other players are fixed to their true values?

In what follows we will fix the values of the other players t_{-i} for some player i. We will concentrate on how a truthful mechanism for the scheduling problem separates the regions with different allocations for player i, when the vector of variables t_i

takes values in ${\mathbb{R}_+}^m$. There are 2^m regions as for each one of the m tasks we have two choices: 1 if player i gets the task and 0 if he doesn't.

We will follow an approach similar to the one in [35] that applies the ideas from [26, 46] on this scheduling problem.

Recall that for every fixed t_{-i}, and any two different assignments a, b for player i we have defined $f^i_{a:b}(t_{-i}) := \sup\{(b-a)\cdot t_i \mid t_i \in R^i_b\}$.

For every player i and every fixed t_{-i}, we define an edge-weighted directed graph $G_{t_{-i}}$, the allocation graph, whose vertex set are all possible allocations of player i. For each two allocations a, b the weight of the edge from a to b is $f^i_{a:b}(t_{-i})$.

It turns out that the difference of payments between two regions R_b and R_a should be bounded from above by the weight of the edge from vertex a to vertex b on the allocation graph. We can see this as follows: the mechanism is truthful so player i does not have an incentive to falsely declare t'_i instead of t_i so $p^i(a_i, t_{-i}) - a_i \cdot t_i \geq p^i(a'_i, t_{-i}) - a'_i \cdot t_i$, which can be rewritten as $p^i(a_i, t_{-i}) - p^i(a'_i, t_{-i}) \geq (a_i - a'_i)\cdot t_i$ and consequently

$$p^i(a_i, t_{-i}) - p^i(a'_i, t_{-i}) \geq \sup\{(a_i - a'_i)\cdot t_i \mid t_i \in R^i_a\} = f^i_{a':a}(t_{-i}). \tag{2.1}$$

The system of all possible inequalities (2.1) is feasible iff the associated network contains no positive length cycle. But by [46] the associated network contains no positive length cycle iff it does not have a positive length cycle of length 2, which is exactly what the Monotonicity Property demands.

Definition 24 (Cycle monotonicity)**.** An allocation algorithm satisfies cycle monotonicity if for every player i, every t_{-i}, every integer K and every cycle $a_1, \ldots, a_K, a_{K+1} = a_1$ on the allocation graph

$$\sum_{k=1}^{K} f^i_{a_k:a_{k+1}} \leq 0.$$

It turns out that if we fix a node a of the allocation graph we can express the payment for any other region $R_{a'}$ in terms of the values $f^i_{a:a'}(t_{-i})$.

In fact in the projections of the mechanism for fixed t_{-i}, what we defined as $f^i_{a_i:a'_i}(t_{-i})$ has yet another interpretation: the hyperplane $f^i_{a_i:a'_i}(t_{-i}) = (a_i - a'_i)\cdot t_i$ separates the regions with assignments a_i, a'_i.

The following Lemma is an essential tool for the proof of Theorem 6 and points out that for two regions $R_a, R_{a'}$ with $Hd(a, a') = 1$ the inequality in (2.1) is al-

ways satisfied with equality. Note however that any other pair of regions might satisfy (2.1) with a strict inequality, which would mean that the two regions do not share a common boundary point.

Lemma 6. *Two regions $R_a, R_{a'}$ that share at least one common boundary point satisfy*

$$p^i(a', t_{-i}) - p^i(a, t_{-i}) = f_{a:a'} = -f_{a':a}.$$

Proof. The proof is an extension of [[46], Proposition 5]. By the premises of the Lemma there exists a t_i such that $t_i \in R_a \cap R_{a'}$. Since $t_i \in R_a$ we have

$$p^i(a', t_{-i}) - p^i(a, t_{-i}) \geq f_{a:a'} \geq (a - a') \cdot t_i$$

and since $t_i \in R_{a'}$ we also have

$$p^i(a, t_{-i}) - p^i(a', t_{-i}) \geq f_{a':a} \geq (a' - a) \cdot t_i,$$

which by the Monotonicity Property, $f_{a:a'} + f_{a':a} \leq 0$, implies

$$(a - a')t_i \geq -f_{a':a} \geq f_{a:a'}.$$

So finally

$$f_{a:a'} \leq p^i(a', t_{-i}) - p^i(a, t_{-i}) \leq -f_{a':a} = f_{a:a'} = (a - a')t_i.$$

□

Remark 5. An alternative definition for $f_{a:a'}$ would be to define it as $p^i(a', t_{-i}) - p^i(a, t_{-i})$. The two definitions coincide when the regions $R_a, R_{a'}$ have a common boundary point, and these are also the pairs of allocations for which we will need this definition.

Lemma 7. *Any cycle on the allocation graph in which each pair of consecutive nodes corresponds to a pair of regions sharing a common boundary point has length zero.*

Proof. By Cycle monotonicity we have $f_{a_1:a_2} + f_{a_2:a_3} + \ldots + f_{a_{s-1}:a_s} \geq 0$. We then follow the reverse direction on the previous cycle and as all pairs of consecutive nodes on the path correspond to pairs of regions sharing a common boundary point, we apply Lemma 6 and get $f_{a_s:a_{s-1}} + \ldots + f_{a_3:a_2} + f_{a_2:a_1} = -f_{a_1:a_2} - f_{a_2:a_3} - \ldots - f_{a_{s-1}:a_s} \leq 0$. Combining the two inequalities we get $f_{a_1:a_2} + f_{a_2:a_3} + \ldots + f_{a_{s-1}:a_s} = 0$. □

2.8 Related work

The scheduling problem on unrelated machines is one of the most fundamental scheduling problems [27, 28]. The problem is NP-complete. Lenstra, Shmoys, and Tardos [36] showed that it can be approximated in polynomial within a factor of 2 but no better than 3/2, unless P=NP.

Nisan and Ronen introduced the mechanism-design version of the problem in the paper that founded the algorithmic theory of Mechanism Design [42, 43]. They showed that the well-known VCG mechanism, which is a polynomial-time algorithm and truthful, has approximation ratio n. They conjectured that there is no deterministic mechanism with approximation ratio less than n. They also showed that no mechanism (polynomial-time or not) can achieve approximation ratio better than 2. We improved it to $1+\sqrt{2}$, in [17, 19] and further to $1+\varphi$ in [31].

Nisan and Ronen [42] also gave a randomized truthful mechanism for two players, that achieves an approximation ratio of 7/4. Mu'alem and Schapira [40] proved a lower bound of $2-\frac{1}{n}$ for any randomized truthful mechanism for n machines and generalized the mechanism in [42] to give a $7n/8$ upper bound. Recently Lu and Yu [37] gave a 1.67-approximation universally truthful randomized algorithm improving it later on [38] to a 1.59-approximation algorithm.

In another direction, [16] showed that no fractional truthful mechanism can achieve an approximation ratio better than $2-1/n$. It also showed that fractional algorithms that treat each task independently cannot do better than $(n+1)/2$ and this bound is tight.

In very recent paper [7] Ashlagi, Dobizinski and Lavi prove a lower bound of n for a special class of mechanisms, which they call "anonymous".

Lavi and Swamy [35] considered the special case of the same problem when the processing times have only two possible values low or high, and devised a deterministic 2-approximation truthful mechanism. Very recently Yu [48] generalized their results constructing a randomized $7(1+\epsilon)$-approximation algorithm for the case when the processing times belong to $[L_j, L_j(1+\epsilon)] \cup [H_j, H_j(1+\epsilon)]$ where $L_j < H_j$ and $\epsilon < 1/16mn$.

Another special case of the problem is the problem on related machines in which there is a single value (instead of a vector) for every machine, its speed. Myerson [41] gave a characterization of truthful algorithms for this kind of problems

(one-parameter problems), in terms of a monotonicity condition. Archer and Tardos [5] found a similar characterization and using it obtained a variant of the optimal algorithm which is truthful (albeit exponential-time). They also gave a polynomial-time randomized 3-approximation mechanism, which was later improved to a 2-approximation, in [2], and very recently to a PTAS by Dhangwatnotai, Dobzinski, Dughmi and Roughgarden [21]. These mechanisms are truthful in expectation. Auletta De Prisco, Penna and Persiano [8] provided a deterministic, monotone $(4 + \epsilon)$ approximation algorithm for the case of constant number of machines m. Andelman, Azar, and Sorani [1] improved this to a FPTAS and additionally gave a 5-approximation algorithm for arbitrary m. Kovács improved the approximation ratio to 3 [32] and to 2.8 [33].

Much more work has been done in the context of combinatorial auctions (see for example [4, 11, 14, 22, 10, 23] and the references within).

Saks and Yu [46] proved that, for mechanism design problems with convex domains of finitely many outcomes, which includes the scheduling problem, a simple necessary monotonicity property of the allocations of different inputs (and without any reference to payments) is also sufficient for truthful mechanisms, generalizing results of [26, 34]. Monderer [39] showed that this result cannot be essentially extended to a larger class of domains. Both these results concern domains of finitely many outcomes. There are however cases, like the fractional version of the scheduling problem, when the set of all possible allocations is infinite. For these, Archer and Kleinberg [3] provided a necessary and sufficient condition for truthfulness which generalizes the results of [46].

Chapter 3

The geometry of truthfulness

3.1 The problem

There exists a simple necessary and sufficient condition for truthfulness in convex domains and a finite number of outcomes, the Monotonicity Property. In single parameter domains, like for example in an auction where there is only one item, monotonicity is exactly the monotonicity we know from calculus and the most practical description of truthfulness we could hope for. The allocation should be a monotone (for the case of auctions an increasing, while for the case of scheduling a decreasing) function of the player's valuation for the item. However for the case of two or more items Monotonicity is a local condition that should be satisfied by any pair of instances of the problem and does not give us any clue about the global picture of the mechanism, when considering the whole space of inputs together. We would instead need a global and more intuitive description, hopefully also practical for proving lower bounds. We replace Monotonicity by a geometrical and global characterization of truthfulness, for the case when the valuations are additive.

Until now such a characterization was known in the context of the scheduling unrelated machines problem only for the easy case of two tasks [19] and it turned out to be a a quintessential element of the characterization proof in [18] and the lower bound in [19]. We believe that our result here can be used for obtaining new lower bounds. The only discouraging fact is that even for the case of 3 tasks the different mechanisms are too many and geometrically complicated.

No matter how many are the players participating in a mechanism, determining

whether a mechanism is truthful boils down to a single-player case. Truthfulness requires that for fixed values of the other players, a player should not be able to increase his utility by lying. Studying the mechanism for fixed values of the other players is like studying a single-player case. Consequently in our setting there is a single player and m different indivisible items (or tasks). The player's type is denoted by the vector $t = (t_1, \ldots, t_m)$, where t_i is the valuation of the bidder for the i-th item/task and the allocation is denoted by $a = (a_1, \ldots, a_m)$ where $a_i \in \{0, 1\}$.

We assume that the bidder has additive valuations and hence the bidder's valuation function when his type is t and his allocation a is $v(a, t) = a \cdot t$. In fact it is easy to see that our results also apply if the valuations are of the form $v(a, t) = \lambda(a \cdot t) + \gamma_a$ for some constants λ, γ_a (we can have one different γ_a for each different allocation a). The reason for this is simple namely these valuations also satisfy the Monotonicity Property and moreover the possible truthful mechanisms for such valuations are like in Figure 4.1 (this would not be the case for valuations with $v(11) = t_1 \cdot t_2$ or $v(11) = 2t_1 + t_2$ as the sloped hyperplane would not be 45°). A mechanism consists of an allocation algorithm a and a payment algorithm p. We make the standard assumption that the utilities are quasilinear, that is the utility of the player is $u(a, t) = v(a, t) - p(a)$.

The allocation part of the mechanism gives a partition of the space $\mathbb{R}^m$ of possible values of a player to 2^m different regions, one for each possible different allocation a of the player. But which are exactly the possible partitions of the space the mechanism creates? This is exactly the question we address in this paper.

We know [46] that a mechanism is truthful if and only if its allocation part satisfies the Monotonicity Property, which is a necessary and sufficient condition for truthfulness, that only involves the allocation part of the mechanism. Consequently by determining the possible partitions of the input space created by the allocation part of the mechanism we will eventually give a characterization of truthfulness.

Our problem can be reformulated as an interesting, simple and fun geometrical problem (forgetting everything about mechanisms and game theory) as follows:

Definition 25 (Geometrical statement of the problem)**.** Suppose you have the m-dimensional cube $[0, 1]^m$. The vector a, formed by the coordinates of each one of the vertices of the cube, is the "allocation"(/label) at this point. Consequently there are 2^m different possible "allocations". We want to give to each one of the points in the interior of the cube one of the possible "allocations" so that the monotonicity

condition $(t - t')(a - a') \leq 0$ is satisfied for each pair of different points t, t' in the cube and their corresponding "allocations" a, a'. Which are the partitions of $\mathbb{R}^m$ that satisfy this property?

As it has already been noticed in [26] in the case of additive valuations the boundaries of the mechanism are hyperplanes of a very specific form, every region created by this partition is a convex polyhedron. In this paper we show exactly which (rather few) polytopes are involved in such a partition. For proving our results we reduce the problem to that of determining the allocation graph of the mechanism, i.e. which of the regions share a common boundary. We can then determine the exact geometrical shape of the mechanism because the hyperplane that separates two regions can be easily derived from the monotonicity property.

Our results apply directly to the scheduling unrelated machines problem giving lower bounds for two very interesting special cases of the problem.

3.1.1 Our Tools

Besides the potential applications of our characterization, we believe that also the method we introduce for studying the allocation graph is of particular interest as it provides a very simple way to handle a very complicated partition of the space.

We propose a new, practical, method for determining all possible allocation graphs and the geometrical shapes of the mechanism: For each region R_a of the mechanism instead of considering its complicated geometrical shape we define a box that contains the region. The signs of distances between parallel to each other boundaries of the mechanism determine whether two of these boxes intersect. If two boxes intersect then the corresponding regions share a common boundary. Alternatively if two boxes intersect then there is an edge between the corresponding edges in the allocation graph. These distances however are not independent from each other. Applying cycle-monotonicity for appropriately chosen zero-length cycles allows us to determine how these constants relate. As boundaries between regions that differ only in one allocation always exist we will concentrate on the subgraph of the allocation graph that consists of the edges corresponding to Hamming distance-1 boundaries. For m tasks it is practical to consider this graph as an m-dimensional hypercube.

3.2 The example of two tasks demonstrates the idea

The idea of our approach is best depicted if we apply it for the easy case of two tasks (for which we already know that the two possible mechanisms are depicted in Figure 4.1). We observe that the two lines ($t_1 = f_{11:01}$ and $t_1 = -f_{00:10}$) that are vertical to the axis t_1 and the two lines ($t_2 = f_{11:10}$ and $t_2 = -f_{01:00}$) that are vertical to the axis t_2 have the same distance (otherwise the sloped line would not be 45°).

Another, purely algebraic and more straightforward way, way to obtain this fact is to just to apply once cycle monotonicity. Taking the cycle $00 \to 01 \to 11 \to 10 \to 00$ we get $f_{00:10}+f_{10:11}+f_{11:01}+f_{01:00} = 0$ or equivalently $f_{11:01}+f_{00:10} = f_{11:10}+f_{00:01}$. If we define $c_{12} := f_{11:01} + f_{00:10}$ (This is the distance between the two lines vertical to the axis t_1.) then by the previous cycle it turns out that the distance between the two lines vertical to the axis t_2, which can be expressed as $f_{11:01} + f_{00:10}$ is also equal to c_{12}. Notice that we did not take two cases and did no drawing.

We are now ready to describe the allocation graph of the mechanism: Region R_{11} is contained in the box $t_1 \leq f_{11:01}, t_2 \leq f_{11:10}$. Region R_{00} is contained in the box $t_1 \geq f_{10:00}, t_2 \geq f_{01:00}$. Regions R_{11} and R_{00} share a common boundary line if and only if the boxes that contain them intersect i.e. if and only if $c_{12} > 0$. (Similarly regions R_{01} and R_{10} share a common boundary line if and only if the boxes that contain them intersect i.e. if and only if $c_{12} < 0$.) That is the sign of c_{12} determines which of the two possible shapes has the mechanism.

Knowing the allocation graph we can then very easily draw the picture of the mechanism. In what follows we generalize this idea to describe the allocation graph and the geometry of a truthful mechanism.

3.3 Definitions

Definition 26. We will define the Hamming Distance $\mathrm{Hd}(a, b)$ between two vectors a, b, as the number of positions in which the two vectors are different.

Lemma 8. *Every region R_a satisfies*

$$R_a \subseteq F_a := \{(t_1, \ldots, t_m) \mid t_1 \leq ((-1)^{a_1} f_{a:(a_{-1},1-a_1)}, \ldots, t_m \leq (-1)^{a_m} f_{a:(a_{-m},1-a_m)}\}.$$

This means that every region R_a is included in a box defined by the boundaries of R_a with all regions R_b such that $\mathrm{Hd}(a,b) = 1$. The proof is immediate by the monotonicity property and the definition of $f_{a:b}$.

We proceed to define some constants that generalize this idea we demonstrated for the case of two tasks. The constant $c_{ij|a_{-\{i,j\}}}$ measures the distance between the separating hyperplanes of the mechanism $t_i = f_{11a_{-\{i,j\}}:01a_{-\{i,j\}}}$ and $t_i = f_{10a_{-\{i,j\}}:00a_{-\{i,j\}}}$, which are two parallel hyperplanes corresponding to Hamming distance 1 boundaries. This constant fully describes the geometry of the mechanism if the allocation of all tasks, except for tasks i, j, is fixed to $a_{-\{i,j\}}$. To provide some intuition why we choose these consider that in a decisive mechanism this would give an asymptotic picture of the mechanism: If the values of only two tasks i, j are allowed to be variables, while the remaining tasks with allocation 1 are fixed to the biggest possible value ($+\infty$) and the tasks with allocation 0 are fixed to the smallest possible value, this constant describes the geometry of the mechanism that allocates tasks i, j.

Definition 27. For all i, j and all possible $m-2$-tuples (/allocations) $a_{-\{i,j\}}$ we define

$$c_{ij|a_{-\{i,j\}}} := f_{11a_{-\{i,j\}}:01a_{-\{i,j\}}} + f_{00a_{-\{i,j\}}:10a_{-\{i,j\}}} = f_{11a_{-\{i,j\}}:10a_{-\{i,j\}}} + f_{00a_{-\{i,j\}}:01a_{-\{i,j\}}} \quad (3.1)$$

But are these constants independent from each other? As the following Lemma shows, the answer is no and the relation between these constants is derived from Cycle Monotonicity.

Lemma 9. *If a mechanism is truthful then the constants $c_{ij|a_{-\{i,j\}}}$ satisfy the following equation:*

$$c_{ij|1a_{-\{i,j,k\}}} - c_{ij|0a_{-\{i,j,k\}}} = c_{ik|1a_{-\{i,j,k\}}} - c_{ik|0a_{-\{i,j,k\}}} \quad (3.2)$$

Proof. (Sketch) We get this from the following cycle (also depicted in Figure 3.2) $f_{111:011} + f_{011:010} + f_{010:000} + f_{000:001} + f_{001:101} + f_{101:100} + f_{100:110} + f_{110:111} = 0$. □

By Lemma 8 each region R_a of the mechanism is contained in a box formed by the separating hyperplanes between R_a and all regions with assignment in Hamming distance 1 from a. If we concentrate on a pair of intersecting regions, then the boxes that contain them have a non-empty intersection. But it is also the other way round :

Lemma 10. *If the boxes corresponding to two regions intersect then the regions share a common boundary hyperplane.*

Proof. Consider the projections of the mechanism for two tasks i, j when the processing times for the rest of the tasks are fixed. Then all other terms of the monotonicity property vanish except for the terms corresponding to tasks i, j. Consequently for fixed values of the other players the mechanism should have one of the two shapes in Figure 4.1.

Suppose that the boxes B_a, B_b corresponding to regions R_a, R_b intersect. Say that a, b differ in the allocation of tasks i, j (possibly also in the allocation of other tasks). Then taking the projection for fixed $t_{-\{i,j\}}$ it is obvious that the two regions share a common boundary. □

We proceed to define $d^i_{a:b}$ as the difference of the Hd-1 boundaries on axis i corresponding to two distinct regions R_a, R_b. We have $d^i_{a:1-a} > 0$ for all $i = 1, \ldots, m$ if and only if regions R_a and R_{1-a} intersect.

Even though the geometry of the mechanism is complicated it turns out that we can derive a general formula for the $d^i_{a:b}$s using now a more complicated zero-length cycle on the allocation graph.

Definition 28. We define the distance $d^i_{a:b} := f_{a:1-a_i,a_{-i}} + f_{b:1-b_i,b_{-i}}$.

Lemma 11. *We have $d^i_{a:b} = d^i_{b:a}$ (symmetry) and $d^i_{a:b} = -d^i_{1-a_i,a_{-i}:1-b_i,b_{-i}}$.*

Proof. Directly from the definition we have $d^i_{a:b} = f_{1-a_i,a_{-i}:a} + f_{1-b_i,b_{-i}:b}$ and $d^i_{1-a_i,a_{-i}:1-b_i,b_{-i}} = f_{a:1-a_i,a_{-i}} + f_{b:1-b_i,b_{-i}}$. Consequently $d^i_{a:b} = -d^i_{1-a_i,a_{-i}:1-b_i,b_{-i}}$. □

Lemma 12. *The distance $d^i_{a:1-a}$ can be expressed as the following sum of constants:*

$$d^i_{a:1-a} := \sum_{j \neq i, j \in \{1,\ldots,m\}} (-1)^{a_i + a_j} c_{ij|b_{-\{i,j\}}},$$

where the k-th coordinate of the allocation b_k is $b_k = \begin{cases} 1 - a_k & \text{if } k < j \\ a_k & \text{if } k > j. \end{cases}$

For example we have

$$d^1_{a:1-a} := (-1)^{a_1+a_2} c_{12|a_{-\{1,2\}}} + (-1)^{a_1+a_3} c_{13|1-a_2 a_{-\{1,2,3\}}} + (-1)^{a_1+a_4} c_{14|1-a_2 1-a_3 a_{-\{1,2,3,4\}}} + \ldots + (-1)^{a_1+a_m} c_{1m|1-a_2 1-a_3 \ldots 1-a_{m-1}}.$$

Note that $d^1_{11:00} = c_{1,2} = c$ so Definition 28 is just an extension of Definition 27.

We include the technical proof of Lemma 12 in the Appendix.

3.4 Characterization of 3-Dimensional mechanisms

3.4.1 Calculating the distances

We believe that the tools we have developed in the preceding section are useful for the study of the allocation graph for an arbitrary number of tasks m. We demonstrate this by using them in order to determine the allocation graphs and the corresponding geometrical shapes a truthful mechanism can take for the case $m = 3$.

For the case of 3 tasks we will apply Lemmas 12 and 11 in order to compute the distances $d^i_{a:1-a}$ with respect to the constants $c_{i,j|a_{-\{i,j\}}}$. For simplicity of notation we will write d^j instead of $d^j_{111:000}$, for $j = 1, 2, 3$ and it turns out that all other distances $d^j_{a:b}$, between regions R_a and R_b, can be expressed using the three distances d^1, d^2, d^3 between regions R_{111} and R_{000}. We define the constant e as

$$e = c_{12|0} - c_{12|1} = c_{13|0} - c_{13|1} = c_{23|0} - c_{23|1}. \tag{3.3}$$

then $c_{12|0} = c_{12|1} + e$ and we can rewrite the equalities in the following way:

$$\begin{aligned}
d^1_{111:000} &= c_{13|1} + c_{12|0} = c_{13|1} + c_{12|1} + e = c_{13|0} + c_{12|0} - e = d^1 \\
d^2_{111:000} &= c_{12|1} + c_{23|0} = c_{12|1} + c_{23|1} + e = c_{12|0} + c_{23|0} - e = d^2 \\
d^3_{111:000} &= c_{13|1} + c_{23|0} = c_{13|1} + c_{23|1} + e = c_{13|0} + c_{23|0} - e = d^3
\end{aligned}$$

$$\begin{aligned}
d^1_{011:100} &= -d^1 \\
d^2_{011:100} &= -c_{12|1} + c_{23|1} = d^3 - d^1 \\
d^3_{011:100} &= -c_{13|1} + c_{23|1} = d^2 - d^1
\end{aligned}$$

$$\begin{aligned}
d^1_{101:010} &= -c_{12|1} + c_{13|1} = d^3 - d^2 \\
d^2_{101:010} &= -d^2 \\
d^3_{101:010} &= = -(d^2 - d^1)
\end{aligned}$$

$$\begin{aligned}
d^1_{110:001} &= -(d^3 - d^2) \\
d^2_{110:001} &= -(d^3 - d^1) \\
d^3_{110:001} &= -d^3
\end{aligned}$$

3.4.2 Properties satisfied by the allocation graph

Lemma 13. *There always exist two regions R_a, R_b in* Hd $= 3$ *such that $d^i_{a:b} \geq 0$ for $i = 1, 2, 3$.*

Proof. Suppose towards a contradiction that the statement of the lemma is not true. Then R_{111}, R_{000} do not share a common boundary. There are three cases: either d^1, d^2, d^3 are all negative, or two of them are negative or one of them is negative. Suppose that $d^1 \leq d^2 \leq d^3$ (the proof for any other relative ranking of the three distances is similar we would just find a different pair of intersecting boxes). Then the three cases are: $d^1 \leq d^2 \leq d^3 < 0$ or $d^1 \leq d^2 < 0 \leq d^3$ or $d^1 < 0 \leq d^2 \leq d^3$. In any of the cases we have $d^1_{011:100} \geq 0, d^2_{011:100} \geq 0, d^3_{011:100} \geq 0$ and consequently regions R_{011}, R_{100} intersect. □

Remark 6. In what follows we will make the assumption that this pair of regions

R_a, R_b in Hd = 3 such that $d^i_{a:b} \geq 0$ for $i = 1, 2, 3$, guaranteed to exist by Lemma 13 are R_{111} and R_{000}.

For any mechanism we present here you can get another truthful mechanism by applying the following rotations: Think of the mechanism as a partition of the cube, if you rotate one of the possible partitions so that the faces of the cube go to faces of the cube after the rotation, you also get a truthful mechanism. The reason is that the slope of the separating hyperplane between two regions only depends on their Hamming Distance, i.e. on the number of tasks on which they differ. The characteristic of the rotation we described is that it respects the Hamming distances.

Lemma 14. *If R_{111} and R_{000} intersect then*

a) if $e < 0$ then at least two of the constants $c_{12|1}, c_{13|1}, c_{23|1}$ are strictly positive,

b) if $e > 0$ then at least two of the constants $c_{12|0}, c_{13|0}, c_{23|0}$ are strictly positive.

Proof. We will deal with the case $e < 0$ (the other case is very similar). Observe the second expression for d^i. Since R_{111} and R_{000} share a common boundary we should have $d^i > 0$ for $i = 1, 2, 3$. Each one of the constants $c_{12|1}, c_{13|1}, c_{23|1}$ appears exactly in two of the three distances and $e < 0$. Suppose towards a contradiction that two of the constants were negative, then at least one of the distances d^i would be negative, contradiction. □

Lemma 15. *If a pair of regions R_a, R_{1-a} share a common Hd-3 boundary then no other pair R_b, R_{1-b} of regions share a common Hd-3 boundary.*

Proof. Suppose for example that R_{111}, R_{000} share a common Hd-3 boundary then this boundary is a hyperplane of the form $t_{11} + t_{12} + t_{13}$ =constant. Consequently the boxes that contain the regions intersect and more specifically all three distances d^1, d^2, d^3 are positive. But as by Lemma 11 $d^1_{011:100} = -d^1$, we have $d^1_{011:100} < 0$ and consequently R_{011}, R_{100} do not intersect on axis t_1 and thus cannot share a common Hd-3 boundary. Similarly as $d^2_{101:010} = -d^2$ regions R_{101}, R_{010} do not intersect on axis t_2 and thus cannot share a common Hd-3 boundary and so on. □

Lemma 16. *If R_{111} and R_{000} share a common boundary and $c_{12|1} > 0, c_{13|1} < 0, c_{23|1} > 0, e < 0$ then we also have $c_{12|0} > 0, c_{13|0} < 0, c_{23|0} > 0$.*

Proof. Since R_{111} and R_{000} share a common boundary $d^1 = c_{13|1} + c_{12|0} > 0$ consequently $c_{12|0} > -c_{13|1} > 0$. As $e = c_{13|0} - c_{13|1} < 0$ we have $c_{13|0} < c_{13|1} < 0$. Finally $d^2 = c_{13|1} + c_{23|0} > 0$ and consequently $c_{23|0} > -c_{13|1} > 0$. □

3.4.3 All possible Mechanisms

Definition 29. A degenerate version of a mechanism M is a mechanism for which some of the constants $c_{ij|0}, c_{ij|1}, d^k_{a:b}$, for some $i, j, k \in \{1, 2, 3\}$ and some allocations a, b, become 0, while all other such constants retain the same sign as in the non-degenerate mechanism.

We will describe the possible shapes of the mechanism when a Hd-3 boundary exists and thanks to Lemma 13 any other mechanism is a degenerate version of a mechanism with a Hd-3 boundary. Summarizing all restrictions to the shape of the mechanism we obtained in the previous section we get the following characterization:

Theorem 12. *The possible truthful mechanisms are the following five possible partitions of the space and all their rotations. (In Figure 3.1 you can see their geometrical shapes.)*

As for any mechanism we give here we also include in our characterization all its rotations, we suppose without loss of generality that R_{111}, R_{000} share a common boundary, that $e < 0$ and that the two constants guaranteed to be positive by Lemma 14 are $c_{12|1} > 0, c_{23|1} > 0$.

1. $c_{12|1} > 0, c_{13|1} > 0, c_{23|1} > 0, \quad c_{12|0} > 0, c_{13|0} > 0, c_{23|0} > 0$
2. $c_{12|1} > 0, c_{13|1} > 0, c_{23|1} > 0, \quad c_{12|0} < 0, c_{13|0} < 0, c_{23|0} < 0$
3. $c_{12|1} > 0, c_{13|1} > 0, c_{23|1} > 0, \quad c_{12|0} < 0, c_{13|0} < 0, c_{23|0} > 0$
4. $c_{12|1} > 0, c_{13|1} > 0, c_{23|1} > 0, \quad c_{12|0} > 0, c_{13|0} < 0, c_{23|0} > 0$
5. $c_{12|1} > 0, c_{13|1} < 0, c_{23|1} > 0, \quad c_{12|0} > 0, c_{13|0} < 0, c_{23|0} > 0.$

3.4.4 Knowing a few distances we can draw the whole mechanism

Now we can fully describe the boxes that contain different regions of the mechanism. Let for simplicity in notation $f_1 := f_{111:011}, f_2 := f_{111:101}, f_3 := f_{111:110}$. In fact given

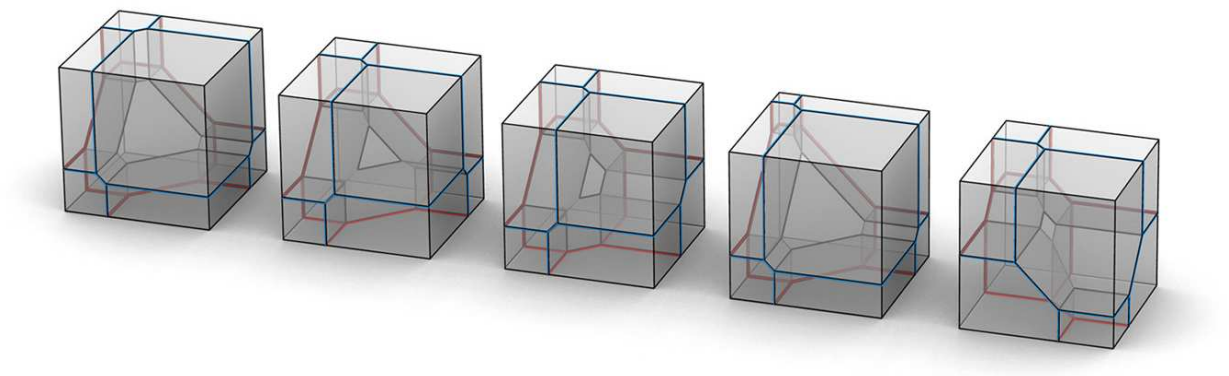

Figure 3.1: 3D models of the possible partitions (up to rotation). Looking just at the blue projections you can determine the constants $c_{ij|0}$ and from the red projections the constants $c_{ij|1}$.

four of the constants $c_{ij|a}$ as input it is easy to find the exact shape of the mechanism thanks to the following relations

$$\begin{aligned}
F_{111} &:= (f_{111:011}, f_{111:101}, f_{111:110}) = (f_1, f_2, f_3) \\
F_{000} &:= (f_1 - d^1, f_2 - d^2, f_3 - d^3) \\
F_{011} &:= (f_1, f_2 - c_{12|1}, f_3 - c_{13|1}) \\
F_{101} &:= (f_1 - c_{12|1}, f_2, f_3 - c_{23|1}) \\
F_{110} &:= (f_1 - c_{13|1}, f_2 - c_{23|1}, f_3) \\
F_{100} &:= (f_1 - d^1, f_2 - c_{23|1}, f_3 - c_{23|1}) \\
F_{010} &:= (f_1 - c_{13|1}, f_2 - d^2, f_3 - c_{13|1}) \\
F_{001} &:= (f_1 - c_{12|1}, f_2 - c_{12|1}, f_3 - d^3).
\end{aligned}$$

Having found a general formula for each one of these boxes we can also describe an algorithm for constructing the geometrical shape of the mechanism. The input is four of the constants $c_{ij|a}$ (knowing these we can compute the other two constants) and the output is the exact shape of the mechanism. We first construct all boxes. Wherever two or more boxes intersect we have to divide the points in the intersection between the regions corresponding to the intersecting boxes. At this point we have to be cautious, because if we are given two intersecting boxes in $\mathrm{Hd} = 3$ then there are infinitely many possible separating hyperplanes between them. However if we

first consider the intersection between Hd = 2 boxes there is a single hyperplane that can separate the two regions. Then consider the intersecting boxes with allocations in Hd = 3 (there is at most one such pair as we will see in lemma 15). As we have already constructed the Hd $\leq$ 2 boundaries there is a single possible Hd-3 separating hyperplane. It is not hard to implement this algorithm and given four of the constants $c_{ij|a}$ of the mechanisms we can fully determine the allocation graph and the geometrical shape of the mechanism. We next proceed to explore which are the possible geometrical shapes of the mechanism, which depend on the signs of the distances.

3.5 Lower bounds for some Scheduling Mechanisms

Observing the figures we got from our characterization we see that many of the regions have the shape of a box, for some of these cases the region that has the shape of the box is $R_{1\ldots1}$. For these cases we can prove a lower bound of $1+\sqrt{n}$ (while the lower bound for the general case is still a small constant).

Theorem 13. *Every mechanism for which $R_{1\ldots1}$ is a box has approximation ratio at least $1+\sqrt{n}$.*

Proof. We will just show that it has ratio at least $\sqrt{n}$. By adding some additional dummy tasks (like in the proof of the $1+\sqrt{2}$ lower bound [17]) it is an easy exercise to improve this to a lower bound of $1+\sqrt{n}$.

Consider the following two $n \times n$ matrices of processing times

$$\begin{pmatrix} b\star & \ldots & b\star \\ 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{pmatrix} \qquad \begin{pmatrix} \epsilon\star & \ldots & b+\epsilon & \ldots & \epsilon\star \\ 1 & \ldots & 1\star & \ldots & 1 \\ \vdots & \ldots & 1 & \ddots & \vdots \\ 1 & \ldots & 1 & \ldots & 1 \end{pmatrix}.$$

There exists some b such that the first player gets all tasks, so that the allocation in the first instance is the one indicated by the stars. If there exists some $b \geq 1$ then the mechanism has approximation n and we are done. Suppose that $0 < b < 1$ and choose b to be the supremum of all these values, the approximation ratio is $n \cdot b$.

There exists a task j such that $f_{1\ldots1:a} = b$ where a differs from $(1, \ldots, 1)$ only in position j. Now consider the second matrix of processing times, since $f_{1\ldots1:a} = b$ and since $R_{1\ldots1}$ has the shape of a box, the allocation is the one indicated by the stars and the approximation ratio is $\frac{1}{b}$. The solution of the equation $n \cdot b = \frac{1}{b}$ is $\sqrt{n}$. □

Finally there is a non-trivial geometrically defined class of mechanisms for which we can provide an n lower bound.

Definition 30. We will say that a mechanism is *non-penalizing* if in the allocation graph no pair of regions of the form R_{a10}, R_{b01}, where a, b are $(m-2)$-dimensional allocation vectors, share a common boundary.

The first mechanism in Figure 3.1 is a example of such a mechanism. The intuition behind these mechanisms is that, if for fixed values of the other players, a player lowers one of his values he only gets more tasks.

Theorem 14. *Every non-penalizing mechanism has approximation ratio at least n.*

The proof uses the same initial instance and is very similar to the proof of a lower bound of n for threshold (/additive) mechanisms.

Proof. For a better understanding we will give the proof of a lower bound of 3 for the case of 3 tasks. Exactly the same technique gives a lower bound of n for $n^2 - n + 1$ tasks and n players as this number of tasks guarantees that one of the players will get at least n tasks.

We start with the instance

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

where we can assume w.l.o.g. that player 1 gets at least 3 tasks. The idea is to lower all values of player 1 to some small $\epsilon > 0$ except for 3 (and in the general case n) values. If we set a task that gets allocated to player 1 to ϵ, then by the Monotonicity Property the allocation of player 1 remains the same. We then lower one by one all of the tasks that are not assigned to player 1. Player 1 doesn't loose any of the tasks initially assigned to him (he might however get more tasks than those initially assigned to him), because the mechanism is non-penalizing mechanism. We continue

until all tasks of player 1 except for the 3 that were initially assigned to him are zero. The approximation ratio is 3.

Example 8. For example if the original assignment is the one marked by the stars we get:

$$\begin{pmatrix} 1\star & 1\star & 1\star & 1 & 1 & 1 & 1\star \\ 1 & 1 & 1 & 1\star & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1\star & 1\star & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1\star & 1\star & 1\star & 0 & 0 & 0 & 0\star \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The tasked whose allocation we do not indicate can be allocated to anyone of the players, but the approximation ratio we get is at least n for any of these allocations.

Exactly the same technique gives a lower bound of n for $n^2 - n + 1$ tasks and n players as this number of tasks guarantees that one of the players will get at least n tasks. □

3.6 Concluding Remarks and open problems

Our characterization is only for the case of 3 tasks, the tools we have developed to obtain this characterization are however for the general case of m tasks. Can we find a succinct way to describe all possible allocation graphs for the general case?

We would like to stress the connection of our results with the scheduling unrelated machines problem. The lower bounds in the last section show that many mechanisms have bad approximation ratio just because of the geometrical shape of their projections. Finally we believe that the characterization for the case of three tasks can be used to improve the existing [31] lower bound of 2.465 for the case of 4 machines to a better constant.

3.7 Some more figures

We have already presented all possible mechanisms, but in this section we will present some of their degenerate versions just in order to make more plausible the notion degeneracy.

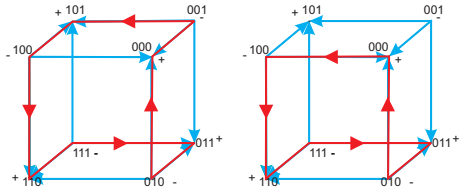

Figure 3.2: The first path gives the equations: $c_{1,2|1} - c_{1,2|0} = c_{1,3|1} - c_{1,3|0}$ and $c_{1,2|1} - c_{1,2|0} = c_{2,3|1} - c_{2,3|0}$. The second path gives the expression for $d^1_{111:000}$.

Remark 7. Actually in the degenerate cases the allocation graph has some additional edges, which are not depicted in Figure 3.4, edges between regions that do not share a full-dimensional boundary. It is very easy to figure out from the geometrical shapes which are these edges but we do not depict them in order to keep the figure easier to understand and more relevant to the geometrical shape.

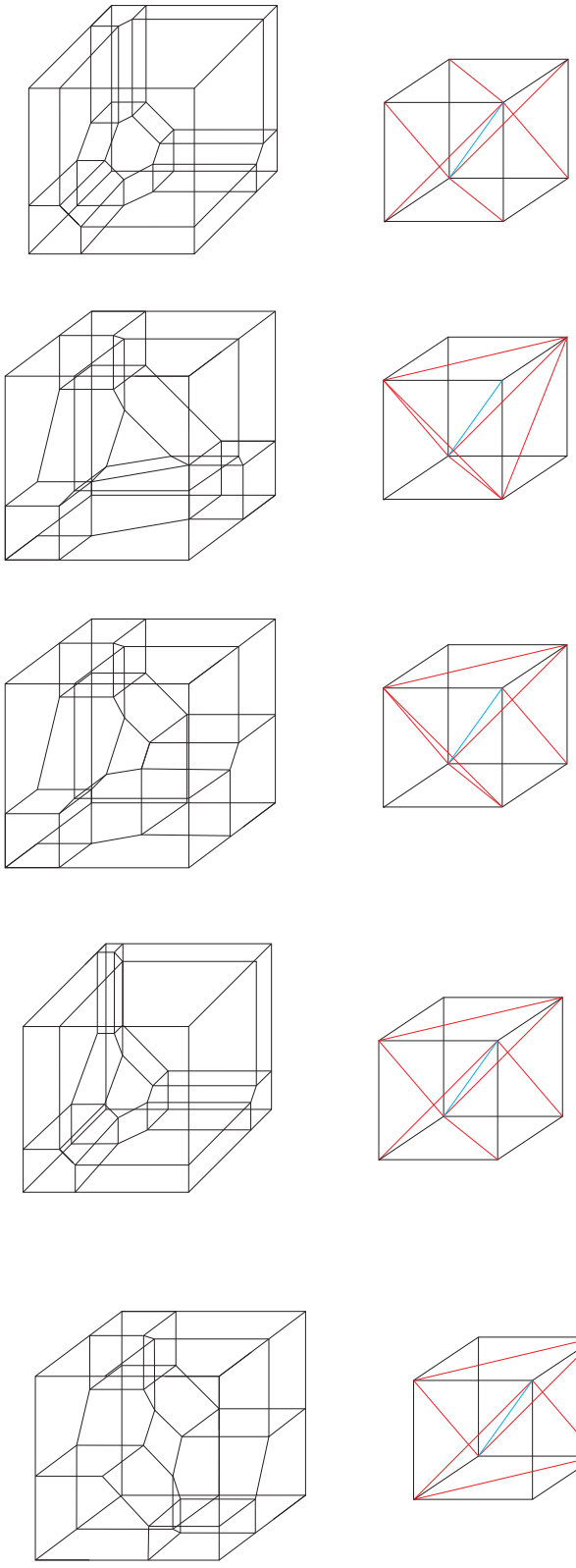

Figure 3.3: Possible (up to rotation) mechanisms with a Hd-3 boundary and all constants $c_{ij|0}, c_{ij|1}, d^k_{a:b} \neq 0$ and the corresponding allocation graph.

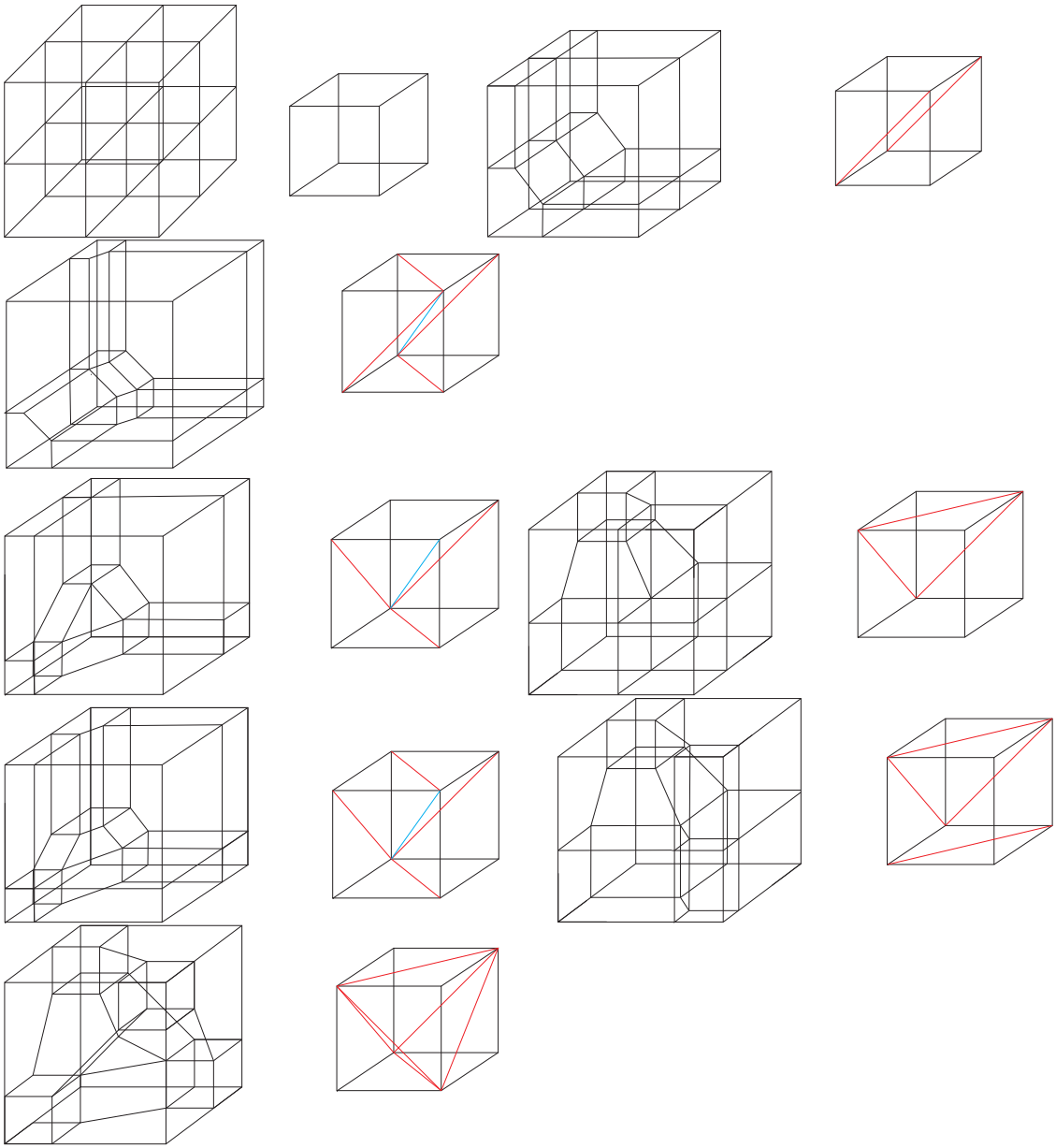

Figure 3.4: Some degenerate mechanisms and their corresponding allocation graph. The Hd-1 boundaries are the edges of the cube. (The edges on the allocation graph corresponding to Hd-2 boundaries are red and the ones corresponding to Hd-3 boundaries blue.)

Chapter 4

A lower bound of $1+\sqrt{2}$

4.1 A geometrical lemma

In our proof of the lower bound, we will exploit the Monotonicity Property of truthful mechanisms. In this section, we present an important lemma that follows from the Monotonicity Property and together with Lemma 4 will be the tools for our proof.

We will also used repeatedly Lemma 4, which states that if a machine gets a set of tasks when it declares t_i, it will get exactly the same set of tasks if we lower the execution time of the tasks allocated to the machine and increase the execution time of the remaining tasks.

It is convenient to allow instances with times $t_{ij} = \infty$. When only finite times are allowed, all the statements are still true; in this case ∞ will simply denote an appropriate arbitrarily high value.

To simplify the presentation, when we apply Lemma 4, we will increase or decrease only some values of a machine, not all its values. The understanding will be that *the rest of the values increase or decrease appropriately by a tiny amount which we omit* to keep the expressions simple.

The second lemma is a useful 2-dimensional property of truthful mechanisms. A useful property that we can extract from Lemma 3, and which is going to play an important role in the proof of our main result, is the following:

Lemma 17. *Fix all values of m tasks except of the values t_{ij} and t_{ik}. Assume that a truthful mechanism assigns both tasks to machine i when $(t_{ij}, t_{ik}) = (1, 0)$ and when $(t_{ij}, t_{ik}) = (0, 1)$. Assume also that the mechanism assigns exactly one of the 2 tasks*

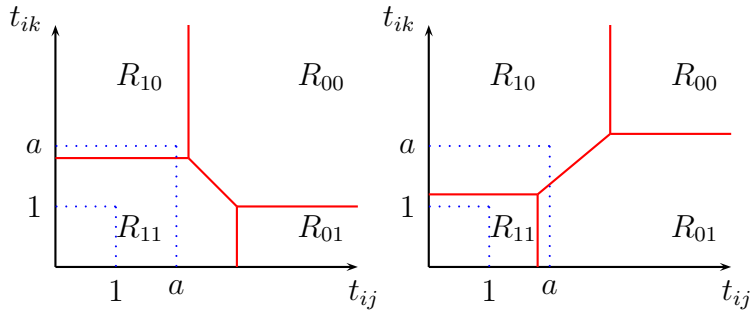

Figure 4.1: Lemma 17.

to machine i when $(t_{ij}, t_{ik}) = (b, b)$ for some $b > 1$. Then the mechanism assigns both tasks to machine i when $(t_{ij}, t_{ik}) = (1, 1)$.

The proof is a simple case analysis and it is essentially shown in Figure 4.1.

4.2 The proof of the main result

We will employ instances with 3 machines and 5 tasks. We will assume throughout that the allocation algorithm does not allocate these values (otherwise the mechanism has arbitrarily high approximation ratio).

The general idea of the proof is the following: We start with the set of tasks

$$t = \begin{pmatrix} 0 & \infty & \infty & b & b \\ \infty & 0 & \infty & b & b \\ \infty & \infty & 0 & b & b \end{pmatrix}$$

where $a > 1$ is a parameter which will be fixed later. This set of tasks has enough symmetries so that it essentially admits two distinct allocations (up to symmetry). For each allocation, we increase or decrease some values appropriately. With the help of the lemmas of the previous section, we show (in Lemma 19 below) that in order to keep the approximation ratio low, the following set of tasks must have the allocation indicated by the stars (in which the first machine gets both tasks 4 and 5):

$$t = \begin{pmatrix} 0\star & \infty & \infty & 1\star & 1\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & \star 0 & b & b \end{pmatrix}.$$

This is sufficient to obtain the lower bound as we will see later.

Lemma 18. *For the instance*

$$\begin{pmatrix} 0 & \infty & \infty & 0 & 1 \\ \infty & 0 & \infty & b & b \\ \infty & \infty & 0 & b & b \end{pmatrix}$$

if the first machine does not get both tasks 4 and 5, then the approximation ratio of the algorithm is at least $1 + b$.

Proof. Suppose that the premises of the lemma hold. As a result, one of machines 2 and 3 will get one of tasks 4 and 5. Suppose without loss of generality that machine 2 gets one of tasks 4 and 5. We raise the 0 of the second player and make it 1 and by Lemma 4 its allocation does not change.

That is, if machine 2 gets task 5, we have

$$\begin{pmatrix} 0\star & \infty & \infty & 0 & 1 \\ \infty & 0\star & \infty & b & b\star \\ \infty & \infty & 0\star & b & b \end{pmatrix} \to \begin{pmatrix} 0\star & \infty & \infty & 0 & 1 \\ \infty & 1\star & \infty & b & b\star \\ \infty & \infty & 0\star & b & b \end{pmatrix},$$

whichever the allocation of the 4th task is (that's what is meant by the absence of a star in the 4th column). Similarly, if machine 2 gets task 4, we have

$$\begin{pmatrix} 0\star & \infty & \infty & 0 & 1 \\ \infty & 0\star & \infty & b\star & a \\ \infty & \infty & 0\star & b & b \end{pmatrix} \to \begin{pmatrix} 0\star & \infty & \infty & 0 & 1 \\ \infty & 1\star & \infty & b\star & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}$$

whichever the allocation of the 5th task is. In either case the cost is at least $1 + a$, while the optimal cost is 1 and is achieved by the allocation

$$\begin{pmatrix} 0\star & \infty & \infty & 0\star & 1\star \\ \infty & 1\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}.$$

□

By symmetry, the previous lemma holds also for the case when the processing times of the first player is $(0, \infty, \infty, 1, 0)$ instead of $(0, \infty, \infty, 0, 1)$.

Lemma 19. *If a truthful mechanism has approximation ratio less than* $1+b$ *then the first machine should get both tasks 4 and 5 of the matrix of processing times*

$$t = \begin{pmatrix} 0 & \infty & \infty & 1 & 1 \\ \infty & 0 & \infty & b & b \\ \infty & \infty & 0 & b & b \end{pmatrix}.$$

Proof. Consider the matrix of processing times

$$t = \begin{pmatrix} 0 & \infty & \infty & b & b \\ \infty & 0 & \infty & b & b \\ \infty & \infty & 0 & b & b \end{pmatrix}.$$

Without loss of generality, the third machine gets none of the tasks 4 and 5. We essentially have two cases.

Case 1: One of machines 1 and 2, suppose without loss of generality that this is machine 1, gets both tasks 4 and 5.

$$\begin{pmatrix} 0\star & \infty & \infty & b\star & b\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}$$

Using Lemma 4, we can lower the values of t_{14} and t_{15} to 1 without changing the allocation. So we have the indicated allocation for the instance

$$\begin{pmatrix} 0\star & \infty & \infty & 1\star & 1\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}.$$

Case 2: Tasks 4 and 5 are allocated to different machines. Without loss of generality, machine 1 gets task 4 (as shown in the first of the three sets of tasks and allocations below). Recall that in the previous lemma (Lemma 18) we showed that the middle matrix of processing times below must have the allocation shown in order to keep the approximation ratio lower than $1+b$. By symmetry, the same is true for the third matrix of processing times below

$$\begin{pmatrix} 0 & \infty & \infty & b\star & b \\ \infty & 0\star & \infty & b & b\star \\ \infty & \infty & 0\star & b & b \end{pmatrix}, \begin{pmatrix} 0\star & \infty & \infty & 0\star & 1\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}, \begin{pmatrix} 0\star & \infty & \infty & 1\star & 0\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}.$$

This is the point in our proof where we consider the geometry of the mechanism. We use Lemma 17 for $i = 1$, $j = 4$, and $k = 5$. The lemma implies that the following set of tasks have the indicated allocation

$$\begin{pmatrix} 0\star & \infty & \infty & 1\star & 1\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix},$$

which proves the lemma. □

We now have all the necessary ingredients to prove our main theorem.

Theorem 15. *There is no deterministic mechanism for the scheduling problem with 3 or more machines with approximation ratio less than* $1 + \sqrt{2}$.

Proof. We will prove that the approximation ratio of any truthful algorithm is at least $\min\{1 + b, 1 + 2/b\}$; for $b = \sqrt{2}$, we have $1+b = 1+2/b$ and the approximation ratio is at least $\min\{1 + b, 1 + 2/b\} = 1 + \sqrt{2}$.

By Lemma 19, in order to have approximation ratio lower than $1 + a$, the allocation of the following matrix of processing times should be as indicated by the stars

$$t = \begin{pmatrix} 0\star & \infty & \infty & 1\star & 1\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}.$$

We can now increase t_{11} to b. By Lemma 4, this does not change the allocation of the first machine. But then for the matrix of processing times and the indicated allocation below

$$\begin{pmatrix} b\star & \infty & \infty & 1\star & 1\star \\ \infty & 0\star & \infty & b & b \\ \infty & \infty & 0\star & b & b \end{pmatrix}$$

the cost is $2+b$, while the optimum cost is b. The approximation ratio is $1+2/b$. Consequently any truthful algorithm has approximation ratio at least $\min\{1 + b, 1 + 2/b\}$.

Of course, if the number of machines is more than 3, the approximation ratio cannot be lower (by setting, for example, all times of the additional machines to ∞). □

Remark 8. By carefully examining Lemma 17 and the proof one can see that if we knew that the mechanism is always like in the first case of Figure 4.1(i.e. if R_{10}, R_{01} are boxes) then we would get a lower bound of $1 + \phi$. This is because in this case a stronger version of Lemma 17 holds namely that: "Fix all values of m tasks except of the values t_{ij} and t_{ik}. Assume that a truthful mechanism assigns both tasks to machine i when $(t_{ij}, t_{ik}) = (1, 0)$ and when $(t_{ij}, t_{ik}) = (0, 1)$. Assume also that the mechanism assigns exactly one of the 2 tasks to machine i when $(t_{ij}, t_{ik}) = (b, b)$ for some $b > 1$. Then the mechanism assigns both tasks to machine i when $(t_{ij}, t_{ik}) = (1, b)$ or when $(t_{ij}, t_{ik}) = (b, 1)$".

In the next section we will eventually get a lower bound of $1 + \phi$ for the general case but using a more involved proof and employing instances with more tasks.

4.3 Lower bound for other L_p norms

The objective in the scheduling problem is to minimize the makespan, i.e. to minimize the maximum completion time, or in other words, to minimize the L_∞ norm of the machine loads. On the other hand the goal achieved by the VCG, which is the best known algorithm, is that of minimizing the sum of completion times, in other words, to minimize the L_1 norm of the machine loads. It turns out that both our lower bound proof can be easily adapted in order to get lower bounds for all L_p norms $2 \leq p < \infty$ where $L_p(t) = \left(\sum_{i=1}^{n} (a_i \cdot t_i)^p\right)^{\frac{1}{p}}$.

Using the proof of the lower bound of $1 + \sqrt{2}$ we get the equation

$$\frac{\left((1+a)^p\right)^{\frac{1}{p}}}{\left(1^p + 1^p\right)^{\frac{1}{p}}} = \frac{\left((2+a)^p\right)^{\frac{1}{p}}}{\left(a^p + a^p + a^p\right)^{\frac{1}{p}}}$$

or equivalently

$$\frac{(1+a)}{2^{\frac{1}{p}}} = \frac{2+a}{3^{\frac{1}{p}}a}.$$

For example if $p = 2$ the equation becomes $\frac{(1+a)}{\sqrt{2}} = \frac{2+a}{a\sqrt{3}}$, which has the solution $a \approx 1.189$. This proves an approximation ration of 1.547 for the objective of minimizing the L_2 norm.

Similarly the proof of the lower bound of $1+\varphi$ gives the equation

$$\frac{\left((1+a)^p\right)^{\frac{1}{p}}}{n^{\frac{1}{p}}} = \frac{\left((2+a)^p\right)^{\frac{1}{p}}}{\left(a^p+a^p+a^p\right)^{\frac{1}{p}}}.$$

Chapter 5

A lower bound of $1+\varphi$

5.1 A lower bound of $1+\varphi$ for $n \to \infty$ machines.

The main result of this section is

Theorem 16. *There is no deterministic mechanism for the scheduling problem with $n \to \infty$ machines with approximation ratio less than $1+\varphi$.*

Moreover for any fixed number of players n, the solution of the equation

$$1 + \frac{1}{b} + \frac{1}{b^2} + \ldots + \frac{1}{b^{n-1}} = 1 + b.$$

is a lower bound for the approximation ratio (Table 5.1.)

We shall build the proof of the theorem around the instance

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \infty & 1 & b & \cdots & b^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & b & b^2 & \cdots & b^{n-1} \\ & & \ddots & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & b^{n-2} & b^{n-1} & \cdots & b^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & b^{n-1} & b^n & \cdots & b^{2n-3} \end{pmatrix},$$

n	2	3	4	5	6	7	8	...	∞
Lower bound	2	2.324	2.465	2.534	2.570	2.590	2.601	...	2.618

Table 5.1: The lower bound given by Theorem 16 for few machines.

where $b \geq 1$ is a parameter and ∞ denotes an arbitrarily high value. Eventually, we will set $b = \varphi$ when $n \to \infty$. We let however b to be a parameter for clarity and for obtaining better bounds for small n.

The lower bound will follow from the fact (which we will eventually prove) that every truthful mechanism with approximation ratio less than $1 + b$ must allocate all $n-1$ rightmost tasks to the first player. The proof of this fact is by induction. However, the induction needs a stronger induction hypothesis which involves instances of the form

$$T(i_1, \ldots, i_k) = \begin{pmatrix} 0 & \infty & \cdots & \infty & b^{i_1} & b^{i_2} & \cdots & b^{i_k} \\ \infty & 0 & \cdots & \infty & b^{i_1+1} & b^{i_2+1} & \cdots & b^{i_k+1} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots \\ \infty & \infty & \cdots & 0 & b^{i_1+n-1} & b^{i_2+n-1} & \cdots & b^{i_k+n-1} \end{pmatrix},$$

where $0 \leq i_1 < i_2 < \ldots < i_k$ are natural numbers and $k \leq n-1$. We allow these instances to have additional tasks for which some value is 0, i.e., additional columns with at least one 0 entry in each one. This is only for technical reasons and will play no significant role in the proof (and it definitely does not affect the optimal cost).

We will call the first n tasks *dummy*. Observe that every mechanism with bounded approximation ratio must allocate the i-th dummy task to player i.

Remark 9. Notice that the optimal allocation for $T(i_1, \ldots, i_k)$ has cost b^{i_k}. Furthermore, if $i_1, i_2, \ldots, i_k$ are all successive natural numbers, then the optimal allocation is unique and coincides with the diagonal assignment. Otherwise there are more than one allocations with optimal cost. For example the allocations indicated by stars:

$$\begin{pmatrix} 0\star & \infty & \infty & \infty & \infty & 1 & b & b^3\star \\ \infty & 0\star & \infty & \infty & \infty & b & b^2\star & b^4 \\ \infty & \infty & 0\star & \infty & \infty & b^2\star & b^3 & b^5 \\ \infty & \infty & \infty & 0\star & \infty & b^3 & b^4 & b^6 \\ \infty & \infty & \infty & \infty & 0\star & b^4 & b^5 & b^7 \end{pmatrix}, \quad \begin{pmatrix} 0\star & \infty & \infty & \infty & \infty & 1 & b & b^3\star \\ \infty & 0\star & \infty & \infty & \infty & b & b^2 & b^4 \\ \infty & \infty & 0\star & \infty & \infty & b^2 & b^3\star & b^5 \\ \infty & \infty & \infty & 0\star & \infty & b^3\star & b^4 & b^6 \\ \infty & \infty & \infty & \infty & 0\star & b^4 & b^5 & b^7 \end{pmatrix}$$

both have the optimal cost b^3.

We will now show the main technical lemma of the proof.

Lemma 20. *Suppose that a truthful mechanism on $T(i_1, \ldots, i_k)$, does not allocate all non-dummy tasks to the first player. Then we can find another instance for which the approximation ratio is at least $1 + b$.*

Proof. Fix a truthful mechanism and suppose that the first player does not get all non-dummy tasks. In the first part, we manipulate the tasks in such a way that we obtain an instance with at least one non-dummy task, whose allocation satisfies the following properties:

- the first player gets no non-dummy task, and
- every other player gets at most one non-dummy task.

In the second part, we show that instances which satisfy the above two conditions, can be changed to obtain an instance with approximation ratio at least $1 + b$.

1st part: Suppose that the first of the above conditions is not satisfied. That is, suppose that the first player gets some non-dummy task. We can then decrease its value (for the first player) to 0. By the Monotonicity Property and in particular by Lemma 4, the same set of tasks will be allocated to the first player, so he still does not get all non-dummy tasks.

Suppose that the second condition is not satisfied, i.e., there is a player in $\{2, \ldots, n\}$ who gets at least two tasks. We can then lower all the non-zero values allocated to this player to 0 except for one. By the Monotonicity Property and in particular by Lemma 4, the same tasks will be allocated to the player. This guarantees that the first player still does not get all non-dummy tasks.

By repeating the above operations, we decrease the number of non-dummy tasks. We will end up with an instance that contains at least one non-dummy task and in which the first player gets no non-dummy task and every other player will get at most one non-dummy task.

Notice that the tasks whose value was changed to 0 remain part of the instance but they will play no particular role in the induction. This is precisely the reason for which we allowed $T(i_1, \ldots, i_k)$ to have additional tasks with at least one 0 entry.

2nd part: We can now assume that there is some $T(i_1, \ldots, i_k)$ with $k \geq 1$ for which the above two conditions are satisfied, i.e, the mechanism allocates no non-dummy task to the first player and at most one non-dummy task to each of the other players.

The optimal cost is b^{i_k}. Our aim is to find a task, which is allocated to some player j, with value at least b^{i_k+1}; we will then increase player j's dummy 0 value

to b^{i_k}. Then by Lemma 4, player j will get both tasks with total value at least $b^{i_k+1} + b^{i_k}$. If the optimal value is still b^{i_k}, then the approximation ratio is at least $1+b$. However, when we raise the dummy 0 to b^{i_k} we may increase the optimal value. The crux of the proof is that there is always an allocated value greater or equal to b^{i_k+1} for which this bad case does not occur. To find such a value we consider two cases:

Case 1: The algorithm assigns a task with value at least b^{i_k+1} to one of the last $n-k$ players. This is the easy case, because we can increase the dummy 0 value of this player to b^{i_k} without affecting the optimum. The reason is that we can allocate the non-dummy tasks to the first k players with cost b^{i_k} (see Remark 9).

Example 9. Consider the following instance with $n = 5$ and $k = 3$. Suppose that the mechanism has the allocation indicated by the stars.

$$\begin{pmatrix} 0\star & \infty & \infty & \infty & \infty & 1 & b & b^3 \\ \infty & 0\star & \infty & \infty & \infty & b & b^2 & b^4\star \\ \infty & \infty & 0\star & \infty & \infty & b^2\star & b^3 & b^5 \\ \infty & \infty & \infty & 0\star & \infty & b^3 & b^4\star & b^6 \\ \infty & \infty & \infty & \infty & 0\star & b^4 & b^5 & b^7 \end{pmatrix}$$

Then we can raise the dummy 0 of the 4-th player to b^3. This does not affect the optimum (which is b^3) but raises the cost of the 4-th player to $b^4 + b^3$.

Case 2: The value of all tasks assigned to the last $n-k$ players is at most b^{i_k}. Consequently the indexes $i_1, i_2, \ldots, i_k$ are not successive integers (Remark 9). Let q be the length of the last block of successive indexes, i.e., $k-q$ is the maximum index where there is a gap in the sequence $i_1, i_2, \ldots, i_k$. More precisely, let $k-q$ be the maximum index for which $i_{k-q}+1 < i_{k-q+1}$. Since player 1 gets no non-dummy task, there is a player $p \in \{q+1, \ldots, n\}$ such that some of the last q tasks is allocated to p. We raise the dummy 0 value of player p to b^{i_k}.

We have to show two properties: Firstly that the value allocated to p was at least b^{i_k+1} and secondly that the optimum cost is not affected. Indeed, the first property follows from the fact that $p > q$ (and by the observation that all values of the last q tasks for the players in $\{q+1, \ldots, n\}$ are at least b^{i_k+1}). To show that the optimal solution is not affected consider the optimal allocation which assigns

- the ℓ-th from the end non-dummy task to the ℓ-player, for $\ell < p$

- the ℓ-th from the end non-dummy task to the $(\ell+1)$-player, for $\ell \geq p$

Notice that this allocation assigns no non-dummy task to the p-th player, as it should. The p-th player is allocated the dummy task, which was raised from 0 to b^{i_k}. Also, since there is a gap in position $k-q$, all allocated values are at most b^{i_k}.

Example 10. Consider the following instance with $n=5$, $k=3$, and $q=2$. Suppose that the mechanism has the allocation indicated by the stars.

$$\begin{pmatrix} 0*\star & \infty & \infty & \infty & \infty & 1 & b^2 & b^3 \\ \infty & 0\star & \infty & \infty & \infty & b & b^3\star & b^4 \\ \infty & \infty & 0\star & \infty & \infty & b^2 & b^4 & b^5\star \\ \infty & \infty & \infty & 0\star & \infty & b^3\star & b^5 & b^6 \\ \infty & \infty & \infty & \infty & 0\star & b^4 & b^6 & b^7 \end{pmatrix}$$

Then $p=3$, and we can raise the dummy 0 of the 3-rd player to b^3. This does not affect the optimum (which allocates the b^3 values), but raises the cost of the 3-rd player to $b^5+b^3 \geq b^4+b^3$.

□

With the above lemma, we can easily prove the main result:

Proof of Theorem 16. Consider the instance

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \infty & 1 & b & \cdots & b^{n-2} \\ \infty & 0 & \cdots & \infty & \infty & b & b^2 & \cdots & b^{n-1} \\ & & \ddots & & & & & & \\ \infty & \infty & \cdots & 0 & \infty & b^{n-2} & b^{n-1} & \cdots & b^{2n-4} \\ \infty & \infty & \cdots & \infty & 0 & b^{n-1} & b^n & \cdots & b^{2n-3} \end{pmatrix}.$$

By the previous lemma, either the approximation ratio is at least $1+a$ or all non-dummy tasks are allocated to the first player. In the latter case, we raise the dummy 0 of the 1-st player to b^{n-1}. The optimal cost becomes b^{n-1} while the cost of the first player is $1+b+b^2+\ldots+b^{n-1}$.

The approximation ratio is at least

$$\min\{1+\frac{1}{b}+\frac{1}{b^2}+\ldots+\frac{1}{b^{n-1}}, b+1\}.$$

We select b so that

$$1 + \frac{1}{b} + \frac{1}{b^2} + \ldots + \frac{1}{b^{n-1}} = 1 + b. \tag{5.1}$$

For $n \to \infty$, this gives

$$\frac{1}{1 - \frac{1}{b}} = 1 + b.$$

Thus $b^2 = 1 + b$, and the solution to this equation is $b = \varphi$. So the approximation ratio of any mechanism is at least $1+\varphi$. For a fixed number of players n, the solution of Equation 5.1 determines a lower bound for the approximation ratio. For small values of n, the approximation ratio is less than $1 + \varphi$ but it converges to it rapidly, as shown in Table 5.1. □

Chapter 6

Characterization of 2-player mechanisms

6.1 Introduction

6.1.1 Do we need the full power of a characterization to get a lower bound?

Seeking a characterization is unrefutably a very natural and important question to ask but can we not prove a lower bound for the problem without getting our hands dirty with a characterization?

There are basically two directions one can follow for providing a lower bound for this problem:

The first approach is to use, an appropriately selected, small subset of the input instances. Fix one instance and consider all its possible allocations (providing a finite approximation ratio). Then argue how each one of the possible allocations results to approximation ratio at least r for some other instance from our chosen set. This is possible because the Monotonicity Property gives a condition that should be satisfied by any two instances of the problem and their corresponding allocations, hence allows us to show how the allocation of one instance affects the allocation of other instances. This approach had been already followed in [42, 17] using a finite set of small instances of 2 and 3 machines respectively and no more than 5 tasks.

Another (more ambitious) approach is to provide a global characterization of

all possible mechanisms, considering all possible inputs, which are infinitely many. After this it is very easy to determine the mechanism with the best approximation ratio. This approach however solves a potentially more difficult problem. The only characterization we know until now, came very recently. For the case of two machines [24] Dobzinski and Sundararajan show that every finite approximation mechanism is task-independent, while in the next section we provide a characterization of all (regardless of approximation ratio) decisive truthful mechanisms in terms of affine minimizers and threshold mechanisms. Until now the only example of a new lower bound obtained by a characterization is the lower bound of 2 for instances with two (or more) tasks [18]. (It is however considerably easier to prove the same lower bound for instances with 3 or more tasks [42] without employing a characterization.)

The proof of the $1 + \varphi$ lower bound does not use a characterization, but in some sense lies somewhere in-between these two different approaches in the sense that it uses an infinite subset of the input and a sophisticated double induction to keep track of how all these allocations depend from each other.

The discouraging thing is that, despite using infinitely more players and tasks, the improvement on the lower bound achieved is very small. This might be considered as an indication that however difficult the characterization approach might be, it is our only serious hope for proving the Conjecture by Nisan and Ronen [42].

6.1.2 What kind of characterization?

As we have already seen in Theorem 1 the allocation of the mechanism to player i is given by the argmin expression $a_i = \text{argmin}_a\{a_i \cdot t_i - p_i(a_i, t_{-i})\}$. The allocations to players must be consistent, i.e., every task is allocated to exactly one machine. The question is what type of allocation algorithms and payment schemes satisfy this property.

There is a simple answer to this question: A mechanism is truthful if and only if it satisfies the *Monotonicity Property*. One nice property of this characterization is that it does not involve the payments at all. Since we usually care about the allocation part of mechanisms, this property focuses exactly on the interesting part. Unfortunately, although this is a necessary and sufficient condition [46], it is not very useful because it is a local and indirect property. The best way to clarify this

point is to consider the case of mechanism design in unrestricted domains. In such domains, the same monotonicity property characterizes the truthful mechanisms. However, there is a much more direct characterization due to Roberts [30]: The class of truthful mechanisms for the unrestricted domain is very limited and contains exactly the class of affine maximizers. An important open problem is to come up with a similar characterization for the scheduling problem and combinatorial auctions. This work resolves this question for 2 players.

For the scheduling problem, very few mechanisms are known to be truthful. The principal example is the VCG mechanism [47, 20, 25] (or second-price mechanism) and its generalization, the affine minimizers [34]. The VCG mechanism allocates each task independently to the machine with minimum value, and pays the machine the second minimum value. VCG can be generalized in two ways and retain its truthfulness.

The first generalization is the task-independent mechanisms, which allocate each task independently of the rest. We know that there are some truthful mechanisms, which are slightly more general; we call them *threshold* mechanisms: For each task j and each player i, there are thresholds h_{ij} such that the player gets the task if and only if the value t_{ij} is less than h_{ij}; the characterizing property of threshold mechanisms is that the threshold depends on the values of the other players, otherwise every mechanism can be expressed with thresholds (see Figure 6.1[$c_1 = 0$] for the geometric fingerprint of these mechanisms which partition the space with orthogonal hyperplanes). For two players we show that the only threshold algorithms are task-independent mechanisms, except for a countable number of inputs (see Examples 15 and 16). For 3 or more players, this doesn't hold and there exist threshold mechanisms, which are far from being task-independent (see Example 11 or [24, subsection 4.3]).

The second generalization of the VCG is the affine minimizers. These can be derived from the VCG in the following way: The VCG selects an allocation that minimizes the sum of processing times, we alter this objective function by applying a linear transformation, we multiply the value of each player by some constant, but more importantly, we alter the value of each allocation by an additive constant. It is this set of additive constants, one per allocation, which make this generalization different than the first generalization.

Both these generalizations are known to be truthful, but they make very poor

algorithms. (For example for the objective of minimizing the makespan, which is the objective of the scheduling problem, we can show a lower bound of n for both of these classes.) The reason is that they allocate each task independently, or almost independently. The question is whether, the affine maximizers and the threshold mechanisms exhaust the class of truthful mechanisms. The answer appears at first to be negative: For example, the mechanism that allocates all tasks to one player, the one with minimum sum of execution times, is truthful but it is neither affine minimizer nor threshold. However, this negative answer is not satisfactory because some allocations are never used, no matter how high or low are the values of the players. (One of the undesired properties of these mechanisms is that they have unbounded approximation ratio.) In contrast, we usually require that mechanisms have a much stronger but very natural property: decisiveness. A mechanism is called *decisive* when a player can enforce an outcome (allocation), by declaring very high or very low values. In fact, for the scheduling problem, it makes more sense to consider *locally-decisive* mechanisms: A player can enforce his allocation by declaring very low or high values, but cannot determine how the remaining tasks are allocated among the other players. When there are only two players, the notions of decisiveness and local-decisiveness coincide, but for 3 or more players decisiveness is a stronger property. We will restrict the discussion in this work to local-decisiveness. In fact, for the case of two tasks our proofs still hold if we only assume decisiveness for 3 allocations (in the sense that a mechanism is decisive for an allocation if each one of the players can impose this allocation by changing his values while the values of the other player remain fixed). This (as well as previous work on similar characterizations [34, 30]) suggest that the right question is to characterize the decisive truthful algorithms. Unfortunately, by restricting our interest to decisive algorithms and positive values, we leave out important truthful specimens because some affine minimizers are not decisive: in some cases, a task will not be allocated to a player even when he declares 0 value for the task. To circumvent this problem, we allow negative values and we characterize the decisive truthful mechanisms for the domain of real values (both positive and negative) (this approach has been also followed in [30]). These algorithms include the affine minimizers and the monotone threshold algorithms; furthermore, every such algorithm is also truthful (but not necessarily decisive) for the nonnegative domain. By allowing negative values, we obtain not only a clean characterization but a useful one too, because we can still

use it to argue about the approximation ratio for nonnegative values.

Our characterization leads us to conjecture:

Conjecture 1. For any number of players, a decisive truthful mechanism partitions the tasks into groups. Each group of tasks is allocated by either an affine minimizer or a threshold mechanism.

Here we show that the conjecture holds for 2 players (Theorem 1). If this turns out to be true for more players, it will show that the class of truthful mechanisms is limited to a few algorithms with poor performance. This will also apply directly to richer domains, such as combinatorial auctions (the richer the domain the more restrictive the class of truthful mechanisms). In fact, for 2 players our theorem is stronger than the conjecture: in the statement of the conjecture, we can replace the threshold mechanisms with the subclass of threshold mechanisms which are task-independent mechanisms, except for countably many points.

Something we should note about the characterization is that each part is not allocated independently of the rest, even for the case of 2 players. Namely the value of a task allocated by an affine minimizer can appear like a "constant" in a threshold mechanism that allocates another task. The following example, shows this for the case of 3 or more players:

Example 11. Consider a mechanism with 3 players and 3 tasks, where the first 2 tasks are allocated by an affine minimizer, while the third task is allocated by a threshold mechanism as follows: the task is given to the first player when he has minimum value ($t_{13} \leq \min\{t_{23}, t_{33}\}$); otherwise it is given to the second player if and only if $t_{23} \cdot t_{11} \leq t_{33}$. Notice that the value t_{11} of the affine minimizer affects the part of threshold mechanisms and more specifically it only affects which of players 2 and 3 gets the task (for these players the value t_{11} is irrelevant and it affects the threshold mechanism in the same way a constant would affect the mechanism).

On a side note, when the affine minimizer in the above mechanism is the VCG mechanism, we obtain an example of a truthful threshold mechanism, which is not task-independent. (After understanding well Example 15 it becomes an easy exercise to construct a similar example for the case of 2 players.)

Recall that for two regions $R_a, R_{a'}$ that share a common boundary

$$f^i_{a:a'}(t_{-i}) = p_i(a'_i, t_{-i}) - p_i(a_i, t_{-i}).$$

For simplicity, we write $f_{a:a'}$ in place of $f^1_{a:a'}$.

Remark 10. In this chapter we also represent the allocations using only the allocation of player 1, since the allocation of player 2 can be inferred. For example, we write $f_{00:10}$ for $f^1_{00:10}$, which is the difference in payments of player 1 when he gets only task 1 and when he gets no task. There is an extra reason to define $f_{a:a'}$: at some point in our proof, we will use the inverse function $f^{-1}_{a:a'}$. The superscript 2 in $f^2_{00:10}$ stands for the second player, mind however that in this chapter (unlike Chapters 2 and 3) the allocations of the subscript however are still allocations of the first player. The corresponding allocations of player 2 can be obtained by changing the roles of 0 and 1.

Another reason for using negative values in our characterization is that the values $f_{a:a'}$, being the differences of payments, can take negative values.

As we mentioned, the allocation of a mechanism can be expressed with argmin expressions, one for every player: $a_i = \text{argmin}_a\{a_i \cdot t_i - p_i(a_i, t_{-i})\}$. For two players and two tasks, we essentially seek the payments that satisfy the following equation, which expresses the fact that the allocations for the two players must be consistent (i.e. each task is allocated exactly once):

$$\begin{aligned}\text{argmin}\{t_{11} + t_{12} - p_1(11, t_2), t_{11} - p_1(10, t_2), t_{12} - p_1(01, t_2), -p_1(00, t_2)\} = \\ \text{argmin}\{-p_2(11, t_1), t_{22} - p_2(10, t_1), t_{21} - p_2(01, t_1), t_{21} + t_{22} - p_2(00, t_1)\}.\end{aligned}$$

Therefore the problem of characterizing the argmin mechanisms for two players and two tasks boils down to the following simple question: Which payments p satisfy the above equation? This is precisely the problem that we are trying to solve here.

The following theorem provides the answer, which applies also to any number of tasks.

We now state our main result:

Theorem 17. *For the scheduling problem with real values every decisive truthful mechanism for 2 players partitions the tasks into groups. Tasks in a group of size at least two are allocated by an affine minimizer and tasks in singleton groups by threshold mechanisms (which are task-independent except for a countable number of inputs).*

The allocation of two different groups is not entirely independent: The values of the tasks in a group allocated by an affine minimizer can appear in the thresh-

old mechanism for a different group of tasks, but only affecting the allocation of a countable number of inputs. The affine minimizers cannot be affected by the values of the tasks in a group allocated by a threshold mechanism.

In fact our characterization does not only hold for additive valuations (like those of the scheduling domain) i.e. $v_i(11) = t_{i1} + t_{i2}, v_i(00) = 0$ but also when the valuations are of the form $v_i(11) = \omega_i(t_{i1} + t_{i2}) + b_i, v_i(00) = 0$, where ω_i, b_i are constants. The reason for this is simple namely these valuations also satisfy the Monotonicity Property and moreover the possible truthful mechanisms for such valuations are like in Figure 6.1 (this would not be the case for valuations with $v_i(11) = t_{i1} \cdot t_{i2}$ or $v_i(11) = 2t_{i1} + t_{i2}$ as the sloped hyperplane would not be 45°.)

6.2 The characterization of decisive mechanisms for 2 tasks

Our main theorem is based on the following theorem which applies to 2 players and 2 tasks and which is the subject of this section.

Theorem 18. *For the scheduling problem with real values the decisive truthful mechanisms for 2 players and 2 tasks are either threshold mechanisms or affine minimizers. The same characterization applies to mechanisms that are decisive for only three outcomes.*

6.2.1 Using a "fix and release" Characterization

For obtaining our characterization we use a new method which we will call a "fix and release" Characterization. We believe that the application of this method might also be of significant help for extending a characterization for the case of more players. This essential part of the proof was missing from the first version of our paper [18]. The idea of this method is to start with the class of all possible mechanisms and then apply a transformation that changes the boundaries of each mechanism in order to make them more simple. This transformation substitutes some of the "irrelevant" variables appearing in the mechanism with arbitrary constants. What we mean by "irrelevant" variable is for example a variable of player 1 affecting which of players 2 and 3 gets a task but not the allocation of player 1 himself like in Example 11.

Then we characterize the restricted class of mechanisms we obtained by applying the transformation. Finally we examine how can the application, of the inverse of any of the possible transformations, affect each one of the mechanisms we obtained from our characterization. This means that we examine how these simple mechanisms are affected if we decide to allow their boundaries to depend in any possible way by "irrelevant variables". The mechanisms we get then are exactly all possible truthful mechanisms. This description seems at first glance too abstract and uncertain to succeed but it becomes more plausible if you examine the following outline of the proof. We proceed in our proof carefully, revealing gradually the properties of $f_{a:a'}$. We assume here that the payments take real (positive or negative) values, so that $f_{a:a'}$ is also a real function.

Proof outline

Suppose that a, a' are two allocations that differ only in one task (in what follows we will deal with the case when they differ in the first task, the case when they differ in the second task is analogous)

- The first step is to show that the function $f_{a:a'}$ is nondecreasing with respect to the variable that changes allocation.
- If $f_{a:a'}$ is continuous as a function of t_{21}, and a, a' differ only in the first task we have $f_{a:a'}(t_{21}, t_{22}) = f_{a:a'}(t_{21}, t'_{22})$ for all $t_{22}, t'_{22} \in \mathbb{R}$.
- Transform $f_{a:a'}$ to a new function depending on a single variable by changing its value at the discontinuities, so that it satisfies the preceding equality for all points. (The function might still have discontinuities, but now only depends on the value of the task that changes allocation.) We show that this transformation is applied to at most countably many points, because the functions $f_{a:a'}$ are nondecreasing and consequently have at most countably many discontinuities.
- Then we can prove that a mechanism with payments of this form is either an affine minimizer or a task-independent mechanism.
- We have to apply the inverse transformation to get all possible mechanisms. This means that we have to change back the payments of the mechanism

types involved in our characterization wherever these are discontinuous and allow them to be affected by the variable that does not change allocation in any possible way (as long as the Monotonicity Property is still satisfied).

- It turns out that by performing the last step the affine minimizers remain intact, while for some of the task-independent mechanisms this transformation allows the variable that does not change allocation (here t_{22}) to affect the mechanism and they become threshold mechanisms, but only for countably many points.

The first step is to show, by means of the Monotonicity Property, that the boundaries between regions that differ only in one task are nondecreasing as functions on the variable that changes allocation (i.e. if we keep the other variable fixed).

Lemma 21. *For a, a' that differ only on one task, the function $f_{a:a'}(t_2)$ is nondecreasing with respect to the variable $(a' - a) \cdot t_2$, when the other variable of the same player, $(a' - a - \mathbf{1}) \cdot t_2$, is fixed.*

For example this means that the function $f_{00:10}(\cdot, t_{22})$ is nondecreasing. The $\cdot$ at the position of t_{21} means that we regard $f_{00:10}$ as a function of t_{21} for fixed t_{22}.

Proof. By symmetry, it suffices to establish the lemma only for the function $f_{00:10}$. Suppose towards a contradiction that it is decreasing for fixed t_{22}. Then there are t_{21}, t'_{21} with $t_{21} < t'_{21}$ and $f_{00:10}(t_{21}, t_{22}) > f_{00:10}(t'_{21}, t_{22})$. The first of the following instances should have the indicated allocation because $\frac{f_{00:10}(t_{21},t_{22})+f_{00:10}(t'_{21},t_{22})}{2} < f_{00:10}(t_{21}, t_{22})$, and applying the Monotonicity Property (for player 2) we get the allocation of the second instance.

$$\begin{pmatrix} \frac{f_{00:10}(t_{21},t_{22})+f_{00:10}(t'_{21},t_{22})}{2} \star & \infty \\ t_{21} & t_{22} \star \end{pmatrix} \to \begin{pmatrix} \frac{f_{00:10}(t_{21},t_{22})+f_{00:10}(t'_{21},t_{22})}{2} \star & \infty \\ t_{21} + \epsilon & t_{22} - \epsilon \star \end{pmatrix}.$$

Similarly, the instances

$$\begin{pmatrix} \frac{f_{00:10}(t_{21},t_{22})+f_{00:10}(t'_{21},t_{22})}{2} & \infty \\ t'_{21} \star & t_{22} \star \end{pmatrix} \to \begin{pmatrix} \frac{f_{00:10}(t_{21},t_{22})+f_{00:10}(t'_{21},t_{22})}{2} & \infty \\ t_{21} + \epsilon \star & t_{22} - \epsilon \star \end{pmatrix}$$

should have the indicated allocation, since for sufficiently small $\epsilon > 0$, $t_{21} + \epsilon < t'_{21}$. This is a contradiction as one instance cannot have two allocations. □

6.2.2 Characterization of mechanisms with simple boundaries.

We will first characterize mechanisms with functions $f_{a:a'}$, (for allocations a, a' that differ only in one task,) that are univariate (depending only on the value of the task that changes allocation).

Property 1 (Univariate Boundaries)**.** We will say that a mechanism satisfies the Univariate Boundaries Property, if for every pair of allocations a, a' that differ only on one task, $f_{a:a'}$ is a univariate function, depending on the value of the task that changes allocation. That is $f_{a:a'}(t_{21}, t_{22}) = g((a'-a)\cdot t_2)$ for some function $g : \mathbb{R} \to \mathbb{R}$. For simplicity, we will write $f_{a:a'}((a'-a)\cdot t_2)$ instead of $g((a'-a)\cdot t_2)$.

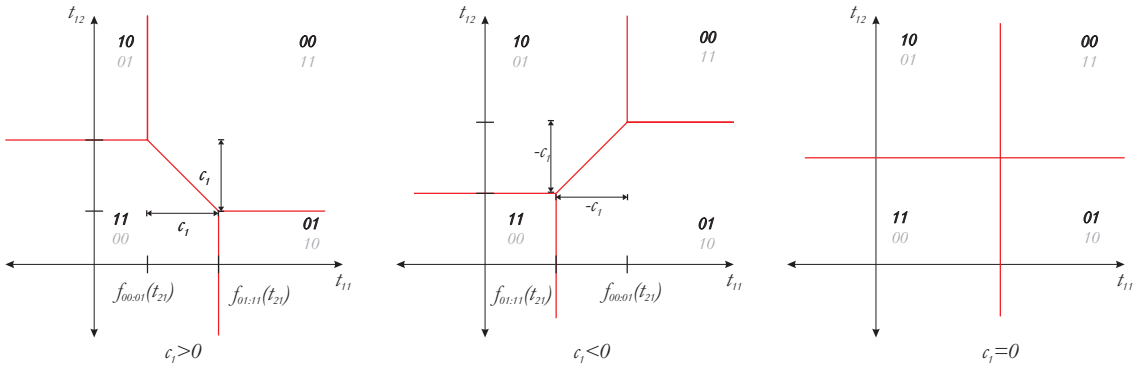

Figure 6.1: There are three ways a truthful mechanism can partition the input space of player 1 for fixed t_2, according to the sign of c_1. For $c_1 = 0$ you can see that there is a threshold $h_j(t_{2j})$ for each task j.

Corollary 1. *The quantities* $c_1 = f_{01:11}(t_2) - f_{00:10}(t_2)$ *and* $c_2 = f^2_{10:00}(t_2) - f^2_{11:01}(t_2)$ *do not depend on* t_2.

Proof. First observe that the following equality

$$f_{01:11}(t_2) - f_{00:10}(t_2) = f_{10:11}(t_2) - f_{00:01}(t_2),$$

follows directly from the definitions because both parts are equal to $p_1(11, t_2) - p_1(01, t_2) - p_1(10, t_2) + p_1(00, t_2)$. The above lemma states that $f_{01:11}(t_2)$ and $f_{00:10}(t_2)$

depend only on t_{21}. Consequently the left part of the above equality depends only on t_{21}. Similarly the right part of the above equality depends only on t_{22}. Therefore, both differences are constant (independent of t_2). We denote this constant by c_1 (the 1 stands for player 1), and we define, in a similar way, a constant c_2 for player 2. □

We can now define the regions of truthful mechanisms. For fixed t_2, let R_{11} denote the set of values t_1 for which the mechanism allocates both tasks to player 1. Region R_{11} which is defined by the following constraints:

$$\begin{aligned} t_{11} &< f_{01:11}(t_{21}) \\ t_{12} &< f_{10:11}(t_{22}) \\ t_{11} + t_{12} &< f_{01:11}(t_{21}) + f_{00:01}(t_{22}). \end{aligned}$$

Similarly, the inequalities for region R_{00} are

$$\begin{aligned} t_{11} &> f_{00:10}(t_{21}) \\ t_{12} &> f_{00:01}(t_{22}) \\ t_{11} + t_{12} &> f_{01:11}(t_{21}) + f_{00:01}(t_{22}). \end{aligned}$$

There are similar constraints that define the other two regions R_{10} and R_{01}. What happens at the boundaries, where the inequality becomes an equality is not determined by the Monotonicity Property. These undetermined values are a major source of difficulty in the characterization of the mechanisms.

From the above inequalities we get that the boundary between regions R_{00} and R_{11}, if it exists, is of the form $t_{11} + t_{12} = f_{01:11}(t_{21}) + f_{00:10}(t_{22})$. Since a similar constraint holds for player 2 (in which the sum $t_{21} + t_{22}$ appears), one could be tempted to conclude that the boundary between allocations 00 and 11 is of the form $t_{11} + t_{12} = h(t_{21} + t_{22})$ for some function h. *Although this conclusion is exactly the one that we will eventually reach, the above argument is fallacious*: There is no justification to identify the boundary between regions R_{00} and R_{11} for the first player when t_2 is fixed and the boundary between the same regions for the second player when t_1 is fixed. In fact, we don't even know that the boundary is some surface when we consider the 4-dimensional space of t. We tried many shortcuts in our proof but we couldn't make them rigorous. This in part is reflected in the writing

of the proof, in which we proceed carefully and use elementary and straightforward arguments.

To proceed to the characterization of mechanisms, we need to understand the functions $f_{00:10}$ and $f_{00:01}$. We have already showed in Lemma 21 that they are non-decreasing.

For most reasonable mechanisms, a stronger statement seems to apply for these two functions: that they are strictly increasing. This however is not generally true as the following example shows.

Example 12 (Task-independent mechanism but not-strictly increasing). Consider the task-independent mechanism with

$$f_{01:11}(t_{21}) = \begin{cases} t_{21} & t_{21} \leq 1 \\ 1 & 1 \leq t_{21} \leq 2 \\ t_{21} - 1 & 2 \leq t_{21} \end{cases}$$

and $f_{00:01}(t_{22}) = t_{22}$. The interesting property of this mechanism is that the function $f_{01:11}(t_{21})$ is not strictly increasing.

But we can show that the functions $f_{01:11}$ and $f_{00:01}$ are indeed strictly increasing when $c_1 \neq 0$. In fact, we show in the next lemma that either the functions are strictly increasing or they are like the following mechanism, which is not a decisive mechanism.

Example 13 (Mechanism with some oblivious player). Consider the mechanism with $f_{00:10}(t_{21}) = b_1$, $f_{00:01}(t_{22}) = b_2$ where b_1, b_2, and c_1 are constants. In this mechanism the first player decides independently of the values of the second player. For given values t_1 of the first player, the second player has the same allocation for every t_2. This mechanism is not decisive, since the second player cannot force all allocations.

Lemma 22. *In a truthful mechanism with $c_1 \neq 0$ the functions $f_{01:11}$ and $f_{00:01}$ are either both strictly increasing or both constant. (The same holds for the pair $f_{00:10}$ and $f_{10:11}$.)*

Proof. We will prove the lemma for $c_1 > 0$ since the case $c_1 < 0$ is very similar. We

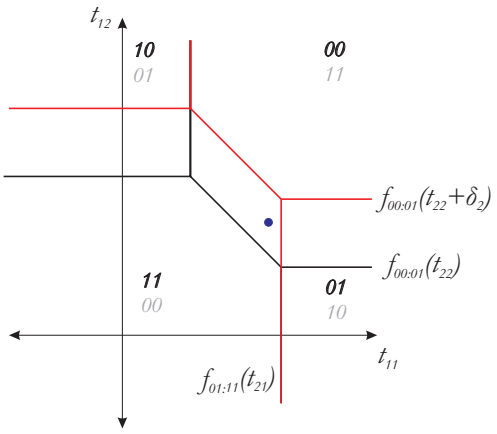

Figure 6.2: The points we use to prove Lemma 22

will show that there are no t_{21}, t_{22}, $0 < \delta_2 < \delta_1$ such that

$$f_{01:11}(t_{21}) = f_{01:11}(t_{21} + \delta_1)$$
$$f_{00:01}(t_{22}) < f_{00:01}(t_{22} + \delta_2)$$

Before we prove this, we show that it implies the lemma. Indeed, if some of the functions $f_{01:11}$ and $f_{00:01}$ is not strictly increasing, say the function $f_{01:11}$, then it is somewhere constant, as we have already established in Lemma 21 that it is non-decreasing. Therefore there are t_{21} and $\delta_1 > 0$ with $f_{01:11}(t_{21}) = f_{01:11}(t_{21} + \delta_1)$. But then for every $\delta_2 < \delta_1$, we must have $f_{00:01}(t_{22}) = f_{00:01}(t_{22} + \delta_2)$. It follows that $f_{00:01}$ is constant. This in turn (with similar reasoning) implies that $f_{10:11}$ is also constant.

We now return to the proof of the above statement. Towards a contradiction we assume that there is such a mechanism with $c_1 > 0$. For $0 < \epsilon < c_1/2$ we consider the inputs (see Figure 6.2)

$$\begin{pmatrix} f_{01:11}(t_{21}) - \epsilon\star & \frac{f_{00:01}(t_{22})+f_{00:01}(t_{22}+\delta_2)}{2} + \epsilon\star \\ t_{21} & t_{22}+\delta_2 \end{pmatrix}$$

and

$$\begin{pmatrix} f_{01:11}(t_{21}) - \epsilon & \frac{f_{00:01}(t_{22})+f_{00:01}(t_{22}+\delta_2)}{2} + \epsilon \\ t_{21}+\delta_1\star & t_{22}\star \end{pmatrix}$$

The claim is that the mechanism will allocate the tasks as indicated by the stars, i. e., both tasks to the first player in the first input and both tasks to the second

player in the second input. Indeed, it is easy to verify that the first input satisfies the inequalities that define R_{11} and the second input satisfies the inequalities that define R_{00}.

But these allocations contradict the Monotonicity Property for player 2. The inputs are identical for the first player while for the second player the sum of the values are $t_{21}+t_{22}+\delta_2$ and $t_{21}+t_{22}+\delta_1$. Since we assumed that $\delta_2 < \delta_1$, the sum of the values of the second player in the first input is less than the sum in the second input. The allocations clearly violate the Monotonicity Property. □

The above lemma establishes that the mechanisms with $c_1 \neq 0$ are either one of the mechanisms of the Example 13 or both functions $f_{01:11}$ and $f_{00:01}$ are strictly increasing. As we consider decisive mechanisms, from now on we will consider only strictly increasing functions.

Lemma 23. *If $c_2 \neq 0$ then the functions $f_{01:11}$ and $f_{00:01}$ are bijections from $\mathbb{R}$ to $\mathbb{R}$.*

Proof. (Recall that the superscript 2 in $f^2_{00:10}$ stands for the second player. By definition, the allocations of the subscript however are still allocations of the first player. The corresponding allocations of player 2 can be obtained by changing the roles of 0 and 1.) We want to establish that the functions $f_{00:10}, f^2_{10:00}$ are inverse. We use the assumption $c_2 \neq 0$ only to guarantee that $f^2_{10:00}$ is strictly increasing.

From the definitions of the function, we get the following implications:

$$\begin{aligned} t_{11} < f_{00:10}(t_{21}) \quad &\Rightarrow \quad f^2_{10:00}(t_{11}) \leq t_{21} \\ t_{11} > f_{00:10}(t_{21}) \quad &\Rightarrow \quad f^2_{10:00}(t_{11}) \geq t_{21} \end{aligned}$$

The claim is that the above conditions imply that the two functions are inverse.

Assume towards a contradiction that for some t_{11} we have $f_{00:10}(f^2_{10:00}(t_{11})) = t'_{11}$ with $t'_{11} > t_{11}$ (the other case, $t'_{11} < t_{11}$, is similar). Then $(t_{11}+t'_{11})/2 > t_{11}$, which by the strictly increasing property of $f^2_{10:00}$ implies that $f^2_{10:00}((t_{11}+t'_{11})/2) > f_{00:10}(t_{11})$. On the other hand, $(t_{11}+t'_{11})/2 < t'_{11} = f_{00:10}(f^2_{10:00}(t_{11}))$ which by the above implications gives $f^2_{01:11}((t_{11}+t'_{11})/2) \leq f^2_{01:11}(t_{11})$, a contradiction. □

The assumption $c_2 \neq 0$ is essential in the above lemma. When $c_2 = 0$, there are mechanisms in which $f_{00:10}$ and $f_{00:01}$ are not bijections; for example, the mechanism of the following example.

Example 14 (Non-decisive task-independent mechanism). Consider the mechanism with $f_{00:10}(t_{21}) = e^{t_{21}}$, $f_{00:01}(t_{22}) = e^{t_{22}}$, and $c_1 = c_2 = 0$. This is a task-independent mechanism which is not decisive; the allocation cannot be expressed as argmin for both players.

Lemma 24. *The constants c_1 and c_2 are either both positive, both negative, or both 0.*

Proof. It suffices to show that if $c_1 > 0$ then it follows that $c_2 > 0$. Consider the tasks

$$\begin{pmatrix} f_{01:11}(t_{21}) - \epsilon\star & f_{00:01}(t_{22}) + 2\epsilon/3\star \\ t_{21} & t_{22} \end{pmatrix} \quad \begin{pmatrix} f_{01:11}(t_{21}) & f_{00:01}(t_{22}) + \epsilon/3 \\ t_{21}\star & t_{22}\star \end{pmatrix}$$

It is straightforward to check the indicated allocations (for $c_1 > \epsilon > 0$).

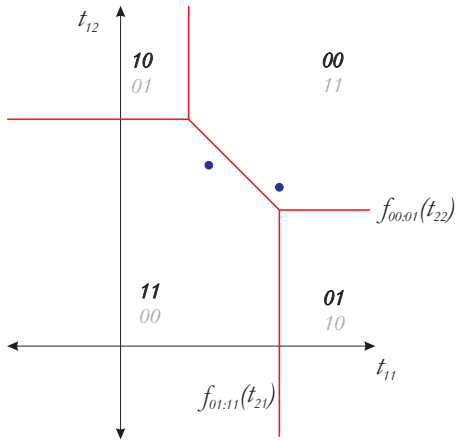

Figure 6.3: The points we take for Lemma 24

Let's denote the above values as: $t_{12} = f_{00:01}(t_{22}) + 2\epsilon/3$ and $t'_{12} = f_{00:01}(t_{22}) + \epsilon/3$. Now, if $c_2 \leq 0$, we should have that $t_{22} \geq f^2_{11:10}(t_{12})$ and $t_{22} \leq f^2_{11:10}(t'_{12}) + c_2$. (Consider the situation player 2 faces when the values of player 1 are fixed to t_1 and t'_1.) But since $t_{12} > t'_{12}$ and since $f^2_{11:10}$ is strictly increasing (as the inverse of a strictly increasing function) this leads to a contradiction when $c_2 \leq 0$. □

We now strengthen the characterization of $f_{00:10}$ and $f_{00:01}$ in the case when $c_1 \neq 0$.

Lemma 25. *For $c_1 \neq 0$, the functions $f_{00:10}$ and $f_{00:01}$ are semiperiodic and in particular they satisfy*

$$f_{00:10}(t_{21} + c_2) = f_{00:10}(t_{21}) + c_1$$

and

$$f_{00:01}(t_{22} + c_2) = f_{00:01}(t_{22}) + c_1.$$

Proof. Again by symmetry we need only to establish the lemma for $f_{00:10}$.

Notice first that $f_{01:11}$ is a bijection, for the same reasons that $f_{00:10}$ is a bijection. We also know that

$$f_{01:11}(t_{21}) = f_{00:10}(t_{21}) + c_1.$$

The associated equation for player 2 is

$$f^{-1}_{00:10}(t_{11}) = f^{-1}_{01:11}(t_{11}) + c_2.$$

We therefore have

$$\begin{aligned} f_{00:10}(t_{21}) + c_1 &= f_{00:10}(f^{-1}_{00:10}(f_{00:10}(t_{21}) + c_1)) \\ &= f_{00:10}(f^{-1}_{01:11}(f_{00:10}(t_{21}) + c_1) + c_2) \\ &= f_{00:10}(f^{-1}_{01:11}(f_{01:11}(t_{21})) + c_2) \\ &= f_{00:10}(t_{21} + c_2) \end{aligned}$$

The first equality is based on the trivial fact that $t_{11} = f_{00:10}(f^{-1}_{00:10}(t_{11}))$. Similarly for the last equality. The second and third equalities follow from the above mentioned equalities for player 2 and player 1. □

We will focus on the case of $c_1 > 0$ as the case $c_1 < 0$ is very similar. Consider the diagonal boundary between the regions R_{11} and R_{00}. This boundary is on the line $t_{11}+t_{12} = f_{01:11}(t_{21})+f_{00:01}(t_{22})$. We have $f_{00:11}(t_{21}, t_{22}) = f_{01:11}(t_{21})+f_{00:01}(t_{22})$. The heart of the characterization is that the function $f_{00:11}(t_{21}, t_{22})$ depends only on the sum of $t_{21} + t_{22}$.

Lemma 26. *The function $f_{00:11}(t_{21}, t_{22}) = f_{01:11}(t_{21}) + f_{00:01}(t_{22})$ depends only on $t_{21} + t_{22}$, i. e., there is some function h such that $f_{00:11}(t_{21}, t_{22}) = h(t_{21} + t_{22})$.*

Proof. Suppose not. That is suppose that there are t_2 and t'_2 such that $t_{21} + t_{22} = t'_{21} + t'_{22}$ and yet $f_{00:11}(t_{21}, t_{22}) < f_{00:11}(t'_{21}, t'_{22})$. If the values differ, they have to differ for some t_{21} and t'_{21} that are very close.

Without loss of generality then we assume that $t_{21} < t'_{21} < t_{21} + c_2$.

This implies that $t'_{22} < t_{22} < t_{22'} + c_2$ and therefore

$$f_{00:01}(t_{22}) < f_{00:01}(t'_{22} + c_2) = f_{00:01}(t'_{22}) + c_1.$$

Let ϵ be a positive parameter with $\epsilon < f_{00:11}(t'_{21}, t'_{22}) - f_{00:11}(t_{21}, t_{22})$ and $\epsilon < f_{01:11}(t'_{22}) - f_{00:01}(t_{22})$. By the above inequalities, ϵ belongs to an open interval and more specifically it can take at least two distinct values. Consider then the values

$$t_{11} = f_{01:11}(t_{21})$$
$$t_{12} = f_{00:01}(t_{22}) + \epsilon$$

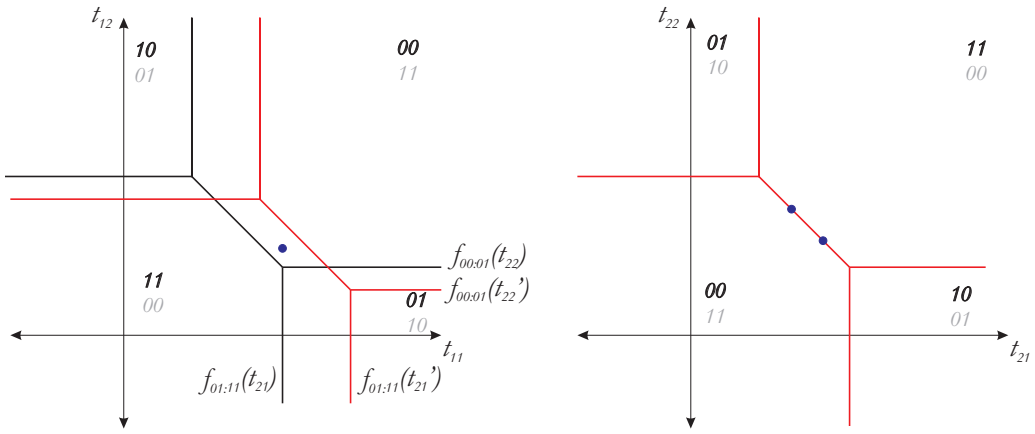

Figure 6.4: The point we use to prove Lemma 26 for player 1 and for player 2.

We can easily verify that the following inputs satisfy the boundary constraints of the appropriate regions (R_{00} and R_{11}) and have the indicated allocations:

$$\begin{pmatrix} t_{11} & t_{12} \\ t_{21}\star & t_{22}\star \end{pmatrix} \quad \begin{pmatrix} t_{11}\star & t_{12}\star \\ t'_{21} & t'_{22} \end{pmatrix}$$

This means that, when we fix t_1, the points t_2 and t'_2 are on the boundary between regions R_{11} and R_{00} of player 2. Equivalently, that

$$t_{21} + t_{22} = f^{-1}_{01:11}(t_{11}) + f^{-1}_{00:01}(t_{12} - \epsilon).$$

(A similar equation holds for t'_2 which however is not different since we assumed that $t_{21} + t_{22} = t'_{21} + t'_{22}$). This equality should hold for every ϵ in some open interval. But this contradicts the fact that $f^{-1}_{00:01}$ is strictly increasing. □

From the last lemma, we get that $h(t_{21} + t_{22}) = f_{01:11}(t_{21}) + f_{00:01}(t_{22})$. We claim that the functions involved are affine as the following lemma shows.

Lemma 27. *If for some real functions h, h_1, h_2 which are continuous at some point, we have $h(x + y) = h_1(x) + h_2(y)$, then all three functions are affine, i. e., they are of the form $ax + b$ for some constants a and b.*

Proof. Let $g(x) = h(x) - h_1(0) - h_2(0)$. We can easily verify that $g(x + y) = g(x) + g(y)$. But this is the Cauchy functional equation. Its only solution is $g(x) = ax$, from which the proof of the lemma follows. □

We have established that the functions $f_{01:11}$ and $f_{00:01}$ are affine but we can say more about their coefficients:

Lemma 28. *When $c_1 \neq 0$, there are constants $\lambda > 0$ and γ_a such that*

$$\begin{aligned} f_{01:11} &= \lambda t_{21} + \gamma_{01} - \gamma_{11}, \\ f_{00:01} &= \lambda t_{22} + \gamma_{00} - \gamma_{01}, \\ f_{00:11} &= \lambda(t_{21} + t_{22}) + \gamma_{00} - \gamma_{11}, \ \textit{for } c_1 > 0, \\ f_{01:10} &= \lambda(t_{21} - t_{22}) + \gamma_{01} - \gamma_{10}, \ \textit{for } c_1 < 0. \end{aligned}$$

Moreover $\lambda = \frac{c_1}{c_2}$.

Proof. The three functions have the same multiplicative constant λ because $f_{00:11}(t_{21} + t_{22}) = f_{01:11}(t_{21}) + f_{00:01}(t_{22})$ and all three functions are linear. It follows that $f_{01:11}(t_{21}) = \lambda t_{21} + \beta$ for some constant β. We can rewrite the constant β as $\gamma_{01} - \gamma_{11}$. Similarly for the other functions.

The fact that $\lambda = \frac{c_1}{c_2}$ follows directly from the linearity and the semiperiodicity of the functions. For example, since $f_{01:11}(t_{21} + c_2) = f_{01:11}(t_{21}) + c_1$ and $f_{01:11}$ is linear it follows that $f_{01:11}(t_{21}) = \frac{c_1}{c_2} t_{21} + \beta$. □

From this and the fact that c_1 and c_2 have the same sign, we get:

Lemma 29. *When $c_1 \neq 0$, the payments of the first player (up to a common additive term which depends on t_2) are of the form $p_1(a_1, t_2) = -\lambda \cdot a_2 \cdot t_2 - \gamma_a$, for some constants $\lambda > 0$ and γ_a.*

With the above payments, the mechanism is the following affine minimizer:

$$\operatorname*{argmin}_a \{a_1 \cdot t_1 + \lambda \cdot a_2 \cdot t_2 + \gamma_a\}.$$

6.2.3 Continuous boundaries can only be univariate functions

Definition 31. (a) We say that a function has a *removable discontinuity* at a point if the one-sided limits of the function at this point are equal to each other (but different than the value of the function).

(b) We say that a function has a *jump discontinuity* at a point if the one-sided limits of the function at this point exist but are different from each other.

It is obvious the discontinuities of a non-decreasing function are all jump-discontinuities. The following Lemma shows that if the value of the function $f^2_{a:a'}$ of player 2 is equal to a constant c for all t_{21} in some open interval (a, b), then the corresponding function $f_{a:a'}$ of player 1 has a discontinuity at point (c, t_{22}) for every t_{22}. This Lemma becomes more clear if you look at Figure 6.2.3.

Lemma 30. *If the function $f^2_{10:00}(\cdot, \infty)$ has constant value, say c, for all $t_{11} \in (a, b)$, then $f_{00:10}(\cdot, t_{22})$ has a jump discontinuity at the point $t_{21} = c$. (Here ∞ denotes a sufficiently large value of t_{12} so that $a_{12} = 0$.)*

Similarly for the other boundaries between allocations that differ in only one task.

Proof. The idea of the proof is that if we know the boundaries of the region with assignment 11 for player 1 that this gives a region where player 2 has assignment 00.

Suppose there exists an interval (a, b), such that for $t_{11} \in (a, b)$ and fixed sufficiently large t_{12} the value of the function $f^2_{10:00}(t_{11}, \infty)$ is a constant, say c. Then for $t_{21} < c, a < t_{11} < b, t_{12} = \infty$ the assignment (of player 1) is 00. Consequently for any fixed t_2 with $t_{21} < c$ the assignment (of player 1) is 00 for $a < t_{11} < b, t_{12} = \infty$,

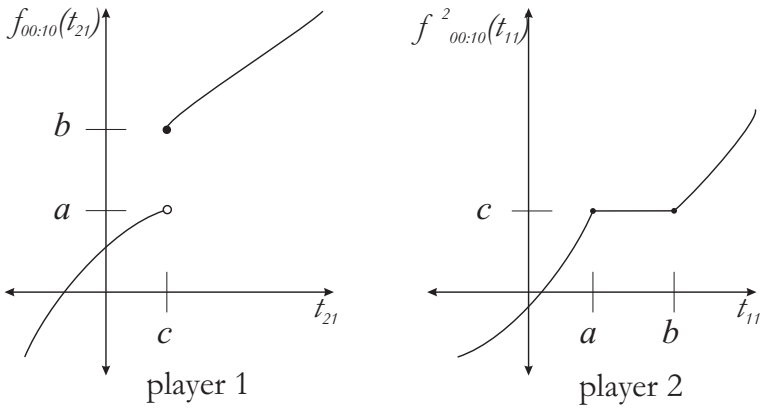

Figure 6.5: If the function of player 1 has a jump discontinuity then the corresponding function of player 2 is somewhere constant

and as the allocation mechanism is monotone, this means that $f_{00:10}(t_{21}, t_{22}) \leq a$ for $t_{21} < c$.

Similarly for $t_{21} > c, a < t_{11} < b, t_{12} = \infty$ the assignment (of player 1) is 10. Consequently for fixed t_2, with $t_{21} > c$ we have that $f_{00:10}(t_{21}, t_{22}) \geq b$. Since $a \neq b$ this shows that for all t_{22}, $f_{00:10}(\cdot, t_{22})$ has a jump discontinuity at $t_{21} = c$. □

Lemma 31. *Take allocations a and a' that differ only in the first task and consider $f_{a:a'}$ as a function of the value t_{21} of the task that changes allocation. If $f_{a:a'}(\cdot, t_{22})$ is continuous at some point t_{21}, then $f_{a:a'}(t_{21}, t_{22}) = f_{a:a'}(t_{21}, t'_{22})$ for all $t_{22}, t'_{22} \in \mathbb{R}$. (An equivalent lemma applies for allocations that differ only on the second task.)*

Consequently the quantity $f_{a:a'}(t_2)$ depends only on $(a - a') \cdot t_2$ (and therefore it depends on only one variable).

Proof. This lemma holds for every number of tasks. We will first prove the lemma for $m = 2$ tasks. We will focus on the case of $a = 00$ and $a' = 10$ since the other cases are very similar.

We will show by contradiction that $f_{00:10}(t_{21}, t_{22})$ does not depend on t_{22}. Suppose that there are t_{21}, t_{22}, and t'_{22} with $t_{22} \neq t'_{22}$ with $f_{00:10}(t_{21}, t_{22}) < f_{00:10}(t_{21}, t'_{22})$.

From the definition of $f_{00:10}(t_{21}, t_{22})$, the tasks of the form

$$\begin{pmatrix} f_{00:10}(t_{21}, t_{22}) + \epsilon & \infty \\ t_{21} \star & t_{22} \star \end{pmatrix}$$

have the indicated allocation for every $\epsilon > 0$, *where* ∞ *indicates an arbitrarily high value* which guarantees that the second task will not be allocated to player 1 (i.e., ∞ is greater than $\max\{f_{00:01}(t_2), f_{00:11}(t_2)\}$).

Similarly, the tasks of the form

$$\begin{pmatrix} f_{00:10}(t_{21}, t'_{22}) - \epsilon \star & \infty \\ t_{21} & t'_{22} \star \end{pmatrix}$$

have the indicated allocation for every $\epsilon > 0$. As we mentioned before, ∞ denotes an arbitrarily high value. We assume of course that the two occurrences of this symbol above denote the same value.

Applying the Monotonicity Property for player 1 we get that for any $t_{11} \in (f_{00:10}(t_{21}, t_{22}) + \epsilon, f_{00:10}(t_{21}, t'_{22}) - \epsilon)$ the tasks of the form

$$\begin{pmatrix} t_{11} & \infty \\ t_{21} \star & t_{22} \star \end{pmatrix}, \begin{pmatrix} t_{11} \star & \infty \\ t_{21} & t'_{22} \star \end{pmatrix}$$

have the indicated allocations.

What follows reveals a very subtle difficulty that arises in the proof of this Lemma. This is the only point where we need the assumption that $f_{00:10}$ has no jump discontinuities. We would like to decrease the values of t_{22} and t'_{22} to $\min\{t_{22}, t'_{22}\}$, and to claim that the allocations remain the same. This is true if player 2 is in the interior of a region, but can be false if he is on a boundary. So we choose t_{11} so that $f^2_{10:00}(t_{11}, \infty) \neq t_{21}$. We can find such a t_{11} because supposing, towards a contradiction, that $f^2_{10:00}(t_{11}, \infty)$ has fixed value t_{21} for t_{11} in the open interval $(f_{00:10}(t_{21}, t_{22}) + \epsilon, f_{00:10}(t_{21}, t'_{22}) - \epsilon)$ then by Lemma 30 $f_{00:10}(\cdot, t_{22})$ is discontinuous at the point t_{21} contradicting our initial assumption. Consequently when the values of player 1 are fixed to (t_{11}, ∞) player 2 is not on the boundary, he is in the interior of a region, which means that the Monotonicity Condition cannot hold with equality. Consequently when we decrease the values of t_{22} and t'_{22} to $\min\{t_{22}, t'_{22}\}$, the allocations remain the same.

This leads to a contradiction because the task

$$\begin{pmatrix} t_{11} & \infty \\ t_{21} & \min\{t_{22}, t'_{22}\} \end{pmatrix}$$

would have two allocations.

The proof can be extended to the case of $m > 2$ tasks: We reduce it to the $m = 2$ case by fixing all tasks except of two. For example, for every $t_2 = (t_{21}, t_{22}, t_{23})$ and $t'_2 = (t_{21}, t'_{22}, t'_{23})$ we have: $f_{000:100}(t_{21}, t_{22}, t_{23}) = f_{000:100}(t_{21}, t'_{22}, t_{23}) = f_{000:100}(t_{21}, t'_{22}, t'_{23})$. □

6.2.4 Extending the characterization to non-continuous functions

Lemma 31 shows that if $f_{a:a'}$ is continuous with respect to the variable that changes allocation, then it is a univariate function, but what if $f_{a:a'}$ is discontinuous?

In fact there exist decisive mechanisms, even with non-infinite approximation ratio, whose boundaries are discontinuous. We give an example of a (threshold) mechanism with boundaries that do not satisfy the conclusion of Lemma 31 and hence also depend on the variable that does not change allocation.

Example 15. Let

$$f_{01:11}(t_{21}, t_{22}) = \begin{cases} t_{21} & t_{21} < 1 \\ 1 & t_{21} = 1, t_{22} \geq 7 \\ 2 & t_{21} = 1, t_{22} < 7 \\ t_{21} + 1 & t_{21} > 1 \end{cases}$$

then the corresponding boundary of player 2 is

$$f^2_{11:01}(t_{11}) = \begin{cases} t_{11} & t_{11} < 1 \\ 1 & 1 \leq t_{11} \leq 2 \\ t_{11} - 1 & t_{11} > 2 \end{cases}$$

Letting $f_{01:11}(t_{21}, t_{22}) = f_{00:10}(t_{21}, t_{22})$ and $f_{10:11}(t_{22}) = f_{00:01}(t_{22}) = t_{22}$ we get a threshold mechanism.

We can generalize this idea to construct a function with an infinite (though countable) number of discontinuities and get a more interesting threshold mechanism that still has non-infinite approximation ratio:

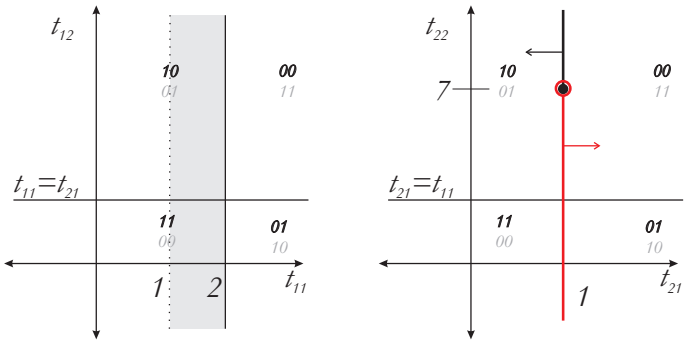

Figure 6.6: The situation in Example 15. Looking at player 2 the allocation cannot be the same along the boundary for getting or not task 1. Notice that the points in the shaded region of player 1 should all have the same allocation for $t_{21} = 1$, which means that for $1 < t_{11} < 2$ and $t_{21} = 1$ player 1 is not on a boundary while player 2 is on a boundary.

Example 16. Let

$$f_{01:11}(t_{21}, t_{22}) = \begin{cases} t_{21} & \text{if } t_{21} < 1 \\ t_{21} + i & \text{if } i < t_{21} < i + 1 \text{ for } i \in \mathbb{N}^* \\ 2i & \text{if } t_{21} = i,\ i \in \mathbb{N}^* \text{ and } t_{22} \text{ prime} \\ 2i - 1 & \text{if } t_{21} = i,\ i \in \mathbb{N}^* \text{ and } t_{22} \text{ non-prime} \end{cases}$$

Letting $f_{01:11}(t_{21}, t_{22}) = f_{00:10}(t_{21}, t_{22})$ and $f_{10:11}(t_{22}) = f_{00:01}(t_{22}) = t_{22}$ we get a threshold mechanism which is not task-independent for an infinite number of inputs, namely for all natural numbers.

However it turns out that thanks to the following classical result in calculus, the discontinuities are few, countably many, so that the conclusion of Lemma 31 holds "almost all of the time".

Theorem 19. *A nondecreasing function cannot have more than countably many discontinuities.*

Consequently $f_{a:a'}(t_2)$, which is non-decreasing can have at most countably many discontinuities as a function of the variable $(a' - a) \cdot t_2$ and fixed value of the other

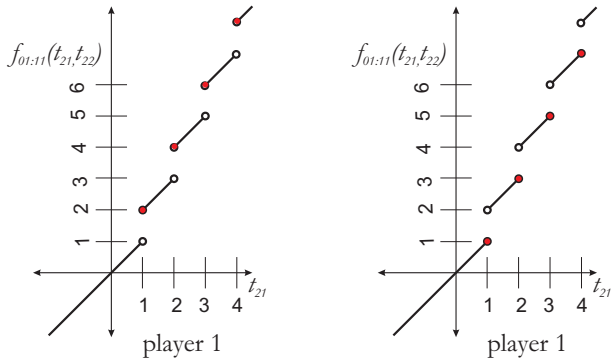

Figure 6.7: The function we define in Example 16 for prime t_{22} is on the left and for non-prime t_{22} is on the right.

variable. Combining the previous theorem with Lemma 31 we get the following Corollary.

Corollary 2. *For allocations a and a' that differ only in the first task, $f_{a:a'}(t_{21}, t_{22}) \neq f_{a:a'}(t_{21}, t'_{22})$ only for countably many points t_{21}.*

Remark 11. Unfortunately for three players there exist threshold mechanisms that are not task-independent for more than countably many inputs.

6.2.5 Transforming $f_{a:a'}$ to a univariate function $f'_{a:a'}$.

Consider all $f_{a:a'}$ between allocations a, a' that differ in only one task. We will do the transformation for $f_{00:10}$ since the transformation for the other functions is similar. If $f_{00:10}(\cdot, t_{21})$ has a jump discontinuity at a point t_{21} and $f_{00:10}(t_{21}, t_{22})$ depends on t_{22} we "change" $f_{00:10}$ to an $f'_{00:10}$ that satisfies $f'_{00:10}(t_{21}, t_{22}) = f'_{00:10}(t_{21}, t'_{22})$ for all $t_{22}, t'_{22} \in \mathbb{R}$ i.e. we do not allow the dependence on the variable that does not change allocation (here t_{22}).

Since $f_{00:10}$ can only have countably many discontinuities as a function of t_{21} $f_{00:10}(t_{21}, t_{22}) \neq f_{00:10}(t_{21}, t'_{22})$ only for countably many points t_{21}, i.e. there exists a function $g : \mathbb{R} \to \mathbb{R}$ such that $f_{00:10}(t_{21}, t_{22}) = g(t_{21})$ except for a countable set D of values for t_{11}, which are all jump discontinuities for g. We define a new function $f'_{00:10}$ with

$$f'_{00:10}(b, t_{22}) = \begin{cases} \lim_{t_{21} \to b^-} g(t_{21}) & \text{for all } b \in D \\ g(b) & \text{else.} \end{cases}$$

(In fact we could alternatively set the value of the function at the discontinuity to any fixed value between $\lim_{t_{21} \to b^-} g(t_{21})$ and $\lim_{t_{21} \to b^+} g(t_{21})$, it just has to be the same value for all t_{22}.) Applying this transformation to all boundaries between regions that differ only in one task and defining the rest of the boundaries $f'_{11:00}, f'_{01:10}$ in the natural way, i.e. $f'_{00:11} := f'_{00:01} + f'_{01:11}$ and $f'_{01:10} := -f'_{00:10} + f'_{00:10}$ we get a new mechanism that satisfies the Univariate Boundaries Property and monotonicity. Consequently applying the characterization we obtained in Section 6.2.2 holds and this mechanism is either an affine minimizer or a threshold mechanism.

6.2.6 Applying the inverse transformation

Starting from the class of all truthful mechanisms and using only the property that some of their boundaries are non-decreasing we applied a transformation that maps this class to the class of truthful mechanisms satisfying the "Univariate Boundaries Property". We managed to characterize this class, so we know the exact form of the boundary functions $f'_{a:a'}$. Which could have been the original class of all truthful mechanisms?

Proof of Theorem 18. Our transformation changed the value of $f_{a:a'}$ only at jump discontinuities. But it is easy to observe that $f_{a:a'}$ has a jump discontinuity if and only if $f'_{a:a'}$ has a jump discontinuity. So the inverse transformation that gives all possible mechanisms alters $f'_{a:a'}$ only at jump discontinuities, allowing the function to depend in any possible way on the variable that does not change allocation, but only at the points where $f_{a:a'}$ is discontinuous.

Starting from an affine minimizer or a **task-independent** mechanism we will now change back countably many points and show that the mechanism becomes either an affine minimizer or a **threshold mechanism**.

Affine minimizer: All functions $f'_{a:a'}$ are strictly increasing linear functions and consequently $f'_{a:a'}$ has no discontinuities. That is applying the inverse of our transformation leaves an affine minimizer intact.

Task-independent mechanism: Suppose $f'_{01:11}$ is discontinuous at a point $t_{21} = b$ and $e_1 = \lim_{t_{21}\to a^-} f'_{01:11}(t_{21}) = f'_{01:11}(b) < \lim_{t_{21}\to a^+} f'_{01:11}(t_{21}) = e_4$.

Which can be $f_{01:11}$ given $f'_{01:11}$? As $f_{01:11}$ is non-decreasing we have $e_1 \leq f_{01:11}(b, t_{22}) \leq e_4$. Then for each different value of t_{22} we can have a different value of $f_{01:11}(b, t_{22})$. Suppose that after applying the inverse transformation $f_{01:11}(b, t_{22})$ takes two distinct values e_1, e_2 such that $e_1 < e_2 < e_3 < e_4$ for $t_{22} = d$ and $t_{22} = d'$ respectively, then

$$f_{01:11}(t_{21}, t_{22}) = \begin{cases} \dots & \\ e_1 & \text{for } t_{21} = b - \epsilon \\ e_2 & \text{for } t_{21} = b, t_{22} = d \\ e_3 & \text{for } t_{21} = b, t_{22} = d' \\ e_4 & \text{for } t_{21} = b + \epsilon \\ \dots & \end{cases}.$$

(Here we should be cautious to take sufficiently small d, d' so that $a_{22} = 1$.) Going now to the picture of player 2 (for fixed t_1) we see that any dependence of $f_{01:11}$ on the values of t_{22} can only affect the allocation across the hyperplane $t_{21} = b$, which is one of the boundaries of the mechanism for player 2 (when the values t_1 of player 1 are fixed). This does not violate the Monotonicity Property.

More specifically for any fixed $t_1 = (t_{11}, t_{12})$ satisfying $e_2 < t_{11} < e_3$ we have that $t_{21} = b$ is a boundary hyperplane as

- for $t_{21} = b - \epsilon$ (and sufficiently small t_{22}) the assignment of player 2 is 11, (as $t_{11} > f_{01:11}(b - \epsilon, t_{22}) = e_1$)
- for $t_{21} = b$ the point (b, d) has assignment 01 and the point (b, d') has assignment 11, (as $t_{11} > f_{01:11}(b, d) = e_2$ and $t_{11} < f_{01:11}(b, d') = e_3$)
- for $t_{21} = b + \epsilon$ (and sufficiently small t_{22}) the assignment is 01 ($t_{11} < f_{01:11}(b + \epsilon, t_{22}) = e_4$).

Observe that for any two different points on the line $t_{21} = b$ the Monotonicity Property is satisfied with equality.

On the other hand if t_{11} does not satisfy $e_2 < t_{11} < e_3$, it is easy to see that the points (b, d) and (b, d') are on the same side of the boundary $t_{11} = f_{01:11}(t_{-1})$

and consequently both have the same assignment, so truthfulness cannot be violated. (Example 15 is a special case of this proof, and perhaps also the best way to understand this proof.)

Consequently applying this transformation to a task-independent we get a threshold mechanism. Actually the mechanism is task-independent except for countably many points for which it is a threshold mechanism. □

6.3 The case of many tasks

The generalization of the characterization to more than two tasks is almost straightforward. Fix a truthful mechanism. For two distinct tasks j_1 and j_2 we will write $j_1 \sim j_2$ when there are some values for the other $m-2$ tasks such that the mechanism restricted to tasks j_1 and j_2 is an affine minimizer (i.e., with the associated constant $c_1 \neq 0$). It should be stressed that we require the mechanism restricted to these two tasks to be an affine minimizer for *some* values of the other tasks, not necessarily for all values, but we are going to see that the two are equivalent.

Our aim is to show that the relation $\sim$ is transitive; since it is clearly symmetric, it essentially partitions the tasks into equivalence classes with the exception that classes of size one are not affine minimizer but task-independent mechanisms. Assume that $j_1 \sim j_2$ and $j_2 \sim j_3$. That is, assume that when we fix some values of the other tasks, the mechanism for tasks j_1 and j_2 is an affine minimizer and when we fix some (not necessarily the same) values of the other tasks the mechanism for tasks j_2 and j_3 is also an affine minimizer, not necessarily with the consistent coefficients. Our aim is to show that the coefficients are consistent. We start with the case of two tasks and then we generalize.

Lemma 32. *When $j_1 \sim j_2$, the payments of player 1 satisfy the following for allocations a and b that agree on all other tasks (i.e., tasks not in $\{j_1, j_2\}$):*

$$p_a(t_2) - p_b(t_2) = \lambda_{j_1,j_2} \cdot (a-b)t_2 + \zeta_{a:b},$$

where $\lambda_{j_1,j_2} > 0$ and $\zeta_{a:b}$ are constants.

Proof. For each task j not in $\{j_1, j_2\}$ we consider inputs with $t_{1j} = \infty$ or $t_{1j} = -\infty$ if $a_j = 0$ or $a_j = 1$, respectively. These inputs have allocations that agree with a

and b on tasks not in $\{j_1, j_2\}$. For a fixed value then of the tasks not in $\{j_1, j_2\}$, the mechanism allocates tasks j_1 and j_2 with an affine minimizer. We therefore have

$$p_a(t_2) - p_b(t_2) = \lambda_{j_1,j_2} \cdot (a - b)t_2 + \zeta_{a:b},$$

where $\lambda_{j_1,j_2} > 0$ and $\zeta_{a:b}$ may depend only on the values of the other tasks.

The crucial observation is that when a and b differ in only one task these are constants (and do not depend on other tasks). It remains to show that this also holds when a and b differ on both tasks. However, if a' is the allocation which differs from a on task j_1, we have that

$$p_a(t_2) - p_{a'}(t_2) = \lambda_{j_1,j_2} \cdot (a - a')t_2 + \zeta_{a:a'}$$
$$p_{a'}(t_2) - p_b(t_2) = \lambda_{j_1,j_2} \cdot (a' - b)t_2 + \zeta_{a':b},$$

from which we get that $\zeta_{a:b} = \zeta_{a:a'} + \zeta_{a':b}$ is also constant since it is equal to the sum of two constants. □

We now generalize this lemma to many tasks.

Lemma 33. *When $j_1 \sim j_2$, $j_2 \sim j_3$, $\ldots$, $j_{k-1} \sim j_k$, the payments of player 1 satisfy the following for allocations a and b that agree on all other tasks (i.e., not in $\{j_1, \ldots, j_k\}$):*

$$p_a(t_2) - p_b(t_2) = \lambda_{j_1,\ldots,j_k} \cdot (a - b)t_2 + \zeta_{a:b},$$

where $\lambda_{j_1,\ldots,j_k} > 0$ and $\zeta_{a:b}$ are constants.

Proof. We show the lemma for $k = 3$ since the generalization is straightforward. We first show that the λ coefficients are equal. We know that

$$p_a(t_2) - p_b(t_2) = \lambda_{j_1,j_2} \cdot (a - b)t_2 + \zeta_{a:b} \tag{6.1}$$
$$p_{\hat{a}}(t_2) - p_{\hat{b}}(t_2) = \lambda_{j_2,j_3} \cdot (\hat{a} - \hat{b})t_2 + \zeta_{\hat{a}:\hat{b}} \tag{6.2}$$

when a and b agree on all tasks not in $\{j_1, j_2\}$ and $\hat{a}$ and $\hat{b}$ agree on all tasks not in $\{j_2, j_3\}$. But the above sets of equations overlap when a and b differ only on task j_2. Therefore $\lambda_{j_1,j_2} = \lambda_{j_2,j_3}$ (we call this constant λ_{j_1,j_2,j_3}).

The proof of the lemma for the ζ terms, is identical to the proof of the previous lemma (the case of two tasks): Let a' be the allocation which differs from a in task j_1. With the same argument as in the previous proof, we conclude that $\zeta_{a:b} = \zeta_{a:a'} + \zeta_{a':b}$. This shows that $\zeta_{a:b}$ is constant. □

The relation $\sim$ is symmetric and transitive and it partitions the tasks into equivalence classes. Suppose for simplicity that all tasks belong to one class. Then the mechanism is an affine minimizer (when there are at least 2 tasks). This follows from the last lemma: Fix $b = 1$, i.e. in b all tasks are allocated to player 1. The payment p_b can be set arbitrarily, so we set it to some arbitrary constant γ_b. Then $p_a(t_2) = \lambda \cdot (a-b) \cdot t_2 + \zeta_{a:b} + p_b(t_2) = -\lambda \cdot a_2 \cdot t_2 - \gamma_a$, where we defined $\gamma_a = -\zeta_{a:b} + \gamma_b$ (a constant) and used $\lambda > 0$ as an abbreviation of $\lambda_{1,\ldots,m}$. Then the allocation for player 1 is given by

$$\operatorname*{argmin}_{a_1}\{a_1 t_1 - p_a(t_2)\} = \operatorname*{argmin}_{a_1}\{a_1 t_1 + \lambda a_2 t_2 + \gamma_a\},$$

with λ and γ_a constants.

The above lemma allows as to partition the tasks so that each part is independent of the other parts. Parts that have 2 or more tasks are affine minimizers. Parts that have only 1 task are not necessarily affine minimizers.

6.4 Lower bound for 2 tasks

Although our characterization involves only decisive mechanisms and negative values, it can be extended directly to show that the approximation ratio even for two tasks is at least 2. The following claim from [24] shows a non-decisive mechanism for positive values has unbounded ratio:

Suppose for example that the allocation 10 does not occur for some t_2, and take the input $\begin{pmatrix} \epsilon & \infty \\ t_{21}\star & t_{22}\star \end{pmatrix}$. Since the allocation of the first player cannot be 10 the allocation is indicated by the stars. By monotonicity the allocation is the same for the instance $\begin{pmatrix} \epsilon & \infty \\ t_{21} - \epsilon\star & \epsilon\star \end{pmatrix}$. But this gives approximation ratio $t_{21}/\epsilon \to \infty$.

The following theorem reproduces the result in [24] for any number $m \geq 2$ of tasks.

Theorem 20. *No truthful mechanism for 2 players with $c_1 \neq 0$ can have a bounded approximation ratio. Consequently any mechanism for 2 players with bounded approximation ratio is a task independent mechanism.*

Proof. The reason is that for small values of t_{21} and t_{22}, the constant c_1 dominates the algebraic expressions of the mechanism and as a result the mechanism is far from optimal.

Towards a contradiction suppose that a mechanism with $c_1 \neq 0$ has bounded approximation ratio r. We essentially look at the one dimensional case. Specifically, consider the case of $t_{12} = \epsilon$. In this way the optimal cost for the second task is almost zero and we can concentrate on the first task. The mechanism gives the first task to player 1 iff $t_{11} \leq f_{00:10}(t_{21}) + c_1$. For the instances with $t_{11} = f_{00:10}(t_{21}) + c_1 \pm \delta$, for some arbitrarily small δ, the approximation ratio is at least $\frac{\max\{t_{21}, f_{00:10}(t_{21})+c_1\}}{\min\{t_{21}, f_{00:10}(t_{21})+c_1\}}$. So we must have

$$t_{21}/r \leq f_{00:10}(t_{21}) + c_1 \leq r t_{21}.$$

Similarly, if we consider the case $t_{22} = \epsilon$, we get that the first player gets the first task iff $t_{11} \leq f_{00:10}(t_{21})$, from which we can conclude that

$$t_{21}/r \leq f_{00:10}(t_{21}) \leq r t_{21}.$$

By subtracting the above inequalities and letting t_{21} to tend to 0, it is clear that the above inequalities cannot hold unless $c_1 = 0$.

(Notice that $f_{00:10}(t_{21})$ is independent of t_{22} because $c_1 \neq 0$.) □

We can now show that even for 2 tasks and 2 players, no mechanism can have approximation ratio less than 2.

Theorem 21. *No mechanism for 2 players and 2 tasks has approximation ratio less than 2.*

Proof. By Theorem 20 the mechanism is task-independent except for countably many points. If we choose a real number l at which both $f_{01:11} = f_{00:10}$ and $f^2_{01:11} = f^2_{00:10}$ are continuous then the mechanism is task-independent(we can do that because there are only countably many points at which one of the two functions is dicontinuous). Suppose that when the processing times of both players for the first task are both l, player 1 gets it. Then the allocation for the instance $\begin{pmatrix} l\star & l\star \\ l & \infty \end{pmatrix}$ is indicated by the stars and the resulting approximation ratio is 2. (In the other case we take the matrix $\begin{pmatrix} l & \infty \\ l\star & l\star \end{pmatrix}$) □

In fact, for 2 tasks, we can show that the only truthful mechanism which achieves approximation ratio 2 is the VCG mechanism.

Theorem 22. *The only truthful mechanism for 2 players with approximation ratio 2 is the VCG mechanism.*

Proof. By symmetry, it suffices to show $f_{01:11}(t_{21}) = t_{21}$. (We will show this for all real numbers except for the possible jump discontinuities of the boundaries which are at most countably many. But since the identity function has no jump discontinuities we will finally get that $f_{01:11}(t_{21}) = t_{21}$ for all $t_{21} \in (0, +\infty)$) Consider the instance

$$\begin{pmatrix} f_{01:11}(t_{21}) - \epsilon \star & t_{21} \star \\ t_{21} & \infty \end{pmatrix}.$$

The second task is allocated to player 1 (otherwise the approximation ratio of the mechanism is infinite) and the first task is also allocated to player 1 as the value of the first player does not exceed the threshold $f_{01:11}(t_{21})$. Therefore, by letting ϵ tend to 0, the approximation ratio is at least

$$\frac{f_{01:11}(t_{21}) + t_{21}}{t_{21}}.$$

This ratio is at most 2, only when $f_{01:11}(t_{21}) \leq t_{21}$. Consider also the instance

$$\begin{pmatrix} f_{01:11}(t_{21}) + \epsilon & \infty \\ t_{21} \star & f_{01:11}(t_{21}) \star \end{pmatrix}.$$

(This is similar to the previous instance, where we exchanged the 2 players). By letting ϵ tend to 0, the approximation ratio is at least

$$\frac{f_{01:11}(t_{21}) + t_{21}}{f_{01:11}(t_{21})}.$$

This ratio is at most 2, only when $f_{01:11}(t_{21}) \geq t_{21}$. In conclusion, the mechanism has approximation ratio at most 2 only when $f_{01:11}(t_{21}) = t_{21}$. □

6.5 Concluding remarks

We gave a characterization of decisive truthful mechanisms. What happens for non-decisive mechanisms? For two tasks we have the following cases:

Mechanisms with some oblivious player A mechanism where one player is decisive and the other is not, as in Example 13.

Mechanisms decisive for only 3 allocations Our proof can be extended to this case and shows that the only mechanisms are affine minimizers. An example is the mechanism with only three allocations: 00, 01, 11 and $f_{01:11}(t_{21}) = t_{21}$, $f_{00:01}(t_{22}) = t_{22}$, $f_{00:11}(t_{21} + t_{22}) = t_{21} + t_{22}$ and $c_1 = \infty$.

Mechanisms with only 2 allocations Consider the mechanism which gives either both tasks to player 1 or both tasks to player 2. It gives both tasks to player 1 iff $t_{11} + t_{12} \geq h(t_{21} + t_{22})$ for some increasing function h. This is a mechanism which is neither affine minimizer nor task independent when h is not linear. (In this case we treat two tasks as a single task, so things are like in a single-parameter domain.)

Our work also gives some direction about how could this characterization be generalized for the case of n players and threshold mechanisms, which as we show show coincide with the additive mechanisms defined in [42], seem to play a central role in the characterization of truthful mechanisms.

Bibliography

[1] Nir Andelman, Yossi Azar, and Motti Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In *22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 69–82, 2005.

[2] Aaron Archer. *Mechanisms for Discrete Optimization with Rational Agents.* PhD thesis, Cornell University, January 2004.

[3] Aaron Archer and Robert Kleinberg. Truthful germs are contagious: A local to global characterization of truthfulness. In *ACM Conference on Electronic Commerce (EC)*, 2008.

[4] Aaron Archer, Christos H. Papadimitriou, Kunal Talwar, and Éva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 205–214, 2003.

[5] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2001.

[6] Aaron Archer and Éva Tardos. Frugal path mechanisms. In *13th Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

[7] Itai Ashlagi, Shahar Dobzinski, and Ron Lavi. An optimal lower bound for anonymous scheduling mechanisms. In *ACM Conference on Electronic Commerce (EC)*, 2009.

[8] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In *21st Annual Symposium on Theoretical Aspects of Computer Science (STACS), volume 2996 of LNCS*, pages 608–619, 2004.

[9] Lawrence M. Ausubel and Paul Milgrom. The lovely but lonely vickrey auction. In *Combinatorial Auctions*, chapter 1, pages 17–40. 2006.

[10] Moshe Babaioff, Ron Lavi, and Elan Pavlov. Mechanism design for single-value domains. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, pages 241–247, 2005.

[11] Yair Bartal, Rica Gonen, and Noam Nisan. Incentive compatible multi unit combinatorial auctions. In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 72–87, 2003.

[12] André Berger, Rudolf Müller, and Seyed Hossein Naeemi. Characterizing incentive compatibility for convex valuations. In *to appear in SAGT*, 2009.

[13] Sushil Bikhchandani, Shurojit Chatterji, Ron Lavi, Ahuva Mu'alem, Noam Nisan, and Arunava Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[14] Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 39–48, 2005.

[15] George Christodoulou. *Gametheoretic study of networks*. PhD thesis, University of Athens, 2006.

[16] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. Mechanism design for fractional scheduling on unrelated machines. In *ICALP*, pages 40–52, 2007.

[17] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1163–1169, 2007.

[18] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A characterization of 2-player mechanisms for scheduling. In *Algorithms - ESA, 16th Annual European Symposium*, pages 297–307, 2008.

[19] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. *Algorithmica*, 2008.

[20] E. Clarke. Multipart pricing of public goods. *Public Choice*, 8:17•33, 1971.

[21] Peerapong Dhangwatnotai, Shahar Dobzinski, Shaddin Dughmi, and Tim Roughgarden. Truthful approximation schemes for single-parameter agents. In *Symposium on Foundations of Computer Science (FOCS)*, 2008.

[22] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 610–618, 2005.

[23] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 644–652, 2006.

[24] Shahar Dobzinski and Mukund Sundararajan. On characterizations of truthful mechanisms for combinatorial auctions and scheduling. In *EC*, 2008.

[25] T. Groves. Incentives in teams. *Econometrica*, 41:617•631, 1973.

[26] Hongwei Gui, Rudolf Müller, and Rakesh V. Vohra. Dominant strategy mechanisms with multidimensional types. In *Computing and Markets*, 2005.

[27] D.S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co. Boston, MA, USA, 1996.

[28] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23(2):317–327, 1976.

[29] Arrow Kenneth. *Social Choice and Individual Values*. John Willey and Sons, Cowles Foundation, 1951.

[30] Roberts Kevin. The characterization of implementable choice rules. *Aggregation and Revelation of Preferences*, pages 321–348, 1979.

[31] Elias Koutsoupias and Angelina Vidali. A lower bound of $1+\phi$ for truthful scheduling mechanisms. In *MFCS*, pages 454–464, 2007.

[32] Annamária Kovács. Fast monotone 3-approximation algorithm for scheduling related machines. In *Algorithms - ESA 2005: 13th Annual European Symposium*, pages 616–627, 2005.

[33] Annamária Kovács. *Fast Algorithms for Two Scheduling Problems.* PhD thesis, Universität des Saarlandes, 2007.

[34] Ron Lavi, Ahuva Mu'alem, and Noam Nisan. Towards a characterization of truthful combinatorial auctions. In *44th Symposium on Foundations of Computer Science (FOCS)*, pages 574–583, 2003.

[35] Ron Lavi and Chaitanya Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In *ACM Conference on Electronic Commerce (EC)*, pages 252–261, 2007.

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

[37] Pinyan Lu and Changyuan Yu. An improved randomized truthful mechanism for scheduling unrelated machines. In *STACS*, pages 527–538, 2008.

[38] Pinyan Lu and Changyuan Yu. Randomized truthful mechanisms for scheduling unrelated machines. In *WINE*, 2008.

[39] Dov Monderer. Monotonicity and implementability. In *ACM Conference on Electronic Commerce (EC)*, 2008.

[40] Ah'uva Mu'alem and Michael Schapira. Setting lower bounds on truthfulness. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1143–1152, 2007.

[41] Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[42] Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC)*, pages 129–140, 1999.

[43] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[44] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[45] Jean-Charles Rochet. A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[46] Michael E. Saks and Lan Yu. Weak monotonicity suffices for truthfulness on convex domains. In *EC*, pages 286–293, 2005.

[47] William Vickrey. Counterspeculations, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[48] Changyuan Yu. Truthful mechanisms for two-range-values variant of unrelated scheduling. *Theoretical Computer Science*, 2009.