Overview

Organization

Introduction
  Motivation
  Introduction to Mechanism Design

The VGC Mechanism
  Vickrey-Groves-Clarke Mechanisms
  A first example: 3 agents are bidding for 2 cars
  A second example: The WAD-Problem

Task Scheduling
  Problem description
  An Upper Bound
  A Lower Bound
Why does the world need Mechanism Design?

- Emergence of the Internet
- Distributed computers act selfish
- Sometimes privately known information only
- Aggregate many people towards a social choice
- Enhance network routing
- Enable fair auctions (Ebay,...)
- Google AdWords™
A definition of Mechanism Design

Mechanism Design

Mechanism Design is a subfield of game-theory. It is the art of designing rules of a game to achieve a specific outcome. Desired outcomes are: **Truthfulness**, individual rationality, budget balance and social welfare.

A simple example

The rules of *german soccer games* have been updated in 1995 to make the game more interesting to spectators. 3 points are given to the winner, 0 points to the looser and 1 point to each team if the game has been drawn. This leads to a more aggressive game.
Mechanism Design in the sense of Nisan and Ronan

We make the following assumptions:

- Participants can choose between different strategies
- A participant has private input data termed its type
- Each participant has a personal preference (utility function)
- Participants try to optimize their utility function
- Such selfish participants are termed agents
- The mechanism tries to achieve a specific outcome using payments given to the agents to direct their decisions
Introduction to Mechanism Design

What is a mechanism Design problem?

Definition (Mechanism Design problem)

- There are \( n \) agents with private types \( t^i \in T^i \)
- The output specification maps to each type vector \( t = t^1 \ldots t^n \) a set of allowed outputs \( o \in O \)
- The agents’ utility is \( u^i = p^i + v^i(t^i, o) \) where \( p^i \) is the payment the mechanism hands to the agent and \( v^i \) is an agent’s valuation given its type and the mechanism’s output.

The mechanism tries to reach a specific outcome (e.g. find the shortest path in a graph). The agents are selfish and optimize their utility functions \( u^i \) (e.g. time needed and money received for performing an action).
What is a Mechanism Design optimization problem?

Definition (Mechanism Design optimization problem)

The output specification is given by a positive real valued objective function $g(t, o)$ and a set of feasible outputs $F$:

- **Exact case:** $\hat{o} = \arg\max_o \{g(t, o)\}$
- **Approx. case:** $\hat{O} = \{o \in F | \forall o' \in F : g(t, o) \geq cg(t, o')\}$
  for a given approximation factor $c < 1$.

where $\hat{o}$ and $\hat{O}$ represent the required outputs.
The mechanism

Definition (A mechanism)

A mechanism \( m = (o, p) \) is composed of an output function \( o() \) and an \( n \)-tuple of payments \( p^1()...p^n() \).

- It defines for each agent \( i \) a family of feasible strategies \( A^i \)
- An Agent can choose to perform any strategy \( a^i \in A^i \)
- The mechanism provides an output function \( o = o(a^1...a^n) \)
- and a payment function \( p^i = p^i(a^1...a^n) \) per agent
The mechanism

Definition (An implementation with dominant strategies)
We say that a mechanism is an **implementation** iff

- For each agent $i$ and each type $t^i$ there exists a **dominant strategy** $a^i \in A^i$.
  (A dominant strategy maximizes an agents utility no matter which strategy each of the other agents has chosen.)
- For each tuple of dominant strategies $a^i \ldots a^n$ the output $o(a^i \ldots a^n)$ satisfies the specification (exact or approximately).
The Revelation Principle

Definition (A Truthful Implementation)
We say that a mechanism is **truthful** iff

- \( \forall_i : A^i = T^i \) (an agent’s strategy is to report its type)
- \( \forall_i : a^i = t^i \) (truth-telling) is a dominant strategy

Definition (A Strongly Truthful Implementation)
We say that a mechanism is **strongly truthful** iff

- \( \forall_i : a^i = t^i \) (truth-telling) is the only dominant strategy
The Revelation Principle

**Lemma (Existence of truthful implementations)**

*If there exists a mechanism that implements a given problem with dominant strategies then there exists a truthful implementation as well.*

**Proof.**

We simulate the strategies with the truthful implementation:

- Given: Mechanism $m = (o, p^1...p^n)$ with dominant strategies $a^i(t^i)$
- We define: Mechanism $\hat{m} = (\hat{o}, \hat{p}^1...\hat{p}^n)$ where $\hat{o}(t^1...t^n) = o(a^1(t^1)...a^n(t^n))$ and $\hat{p}^i(t^1...t^n) = p^i(a^1(t^1)...a^n(t^n))$
Vickrey-Groves-Clarke Mechanisms

- The VGC mechanism is one of the most important positive results in mechanism design.
- It is a generalization of the famous Vickrey auction where bidders submit written bids without knowing the bid of the other people in the auction. In the Vickrey auction the highest bidder wins, but the price paid is the second highest bid.
- The VGC mechanism applies to mechanism design maximization problems where the objective function is simply the sum of all agents’ valuations: \( g(t, o) = \sum_i v^i(t^i, o) \)
Utilitarian functions

**Definition (Utilitarians)**

We say that a maximization mechanism design problem is utilitarian if its objective function satisfies $g(t, o) = \sum_i v^i(t^i, o)$.

**Definition (The VGC family)**

A mechanism $m(o(t), p(t))$ belongs to the VGC family iff

- $o(t) \in \text{argmax}_o (\sum_{i=1}^n v^i(t^i, o)) = \text{argmax}_o g(t, o)$
- $p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) + h_i(t^{-i})$

where $h_i$ is an arbitrary function of $t^{-i}$. 
VGC’s truthfulness

Theorem (Groves (1973))

A VGC mechanism is truthful.
Proving VGC’s truthfulness

Proof.
Let us assume that truth-telling is not a dominant strategy!
⇒ There exists a \( \tilde{d}^i \) so that the following holds (*):

\[
g(t, o(d^{-i}, t^i)) = \sum_{j=1}^{n} v^j(t^j, o(d^{-i}, t^i)) \\
= v^i(t^i, o(d^{-i}, t^i)) + \sum_{j=1, j \neq i}^{n} v^j(t^j, o(d^{-i}, t^i)) \\
= v^i(t^i, o(d^{-i}, t^i)) + p^i(d^{-i}, t^i) - h^i(d^{-i}) \\
<^* v^i(t^i, o(d^{-i}, \tilde{d}^i)) + p^i(d^{-i}, \tilde{d}^i) - h^i(d^{-i}) \\
= v^i(t^i, o(d^{-i}, \tilde{d}^i)) + \sum_{j=1, j \neq i}^{n} v^j(t^j, o(d^{-i}, \tilde{d}^i)) \\
= \sum_{j=1}^{n} v^j(t^j, o(d^{-i}, \tilde{d}^i)) \\
= g(t, o(d^{-i}, \tilde{d}^i))
\]

⇒ Contradiction to: \( o(t) \in \text{argmax}_o g(t, o) \)
Weighted VGCs

Definition (Weighted Utilitarian)
We say that a maximization mechanism design problem is weighted utilitarian if its objective function satisfies \( g(t, o) = \sum_i \beta^i v^i(t^i, o) \).

Definition (The weighted VGC family)
A mechanism \( m(o(t), p(t)) \) belongs to the weighted VGC family iff

\[
\begin{align*}
\circ \quad & o(t) \in \arg \max_o \left( \sum_{i=1}^n v^i(t^i, o) \right) = \arg \max_o g(t, o) \\
\circ \quad & p^i(t) = \frac{1}{\beta^i} \sum_{j \neq i} \beta^j v^j(t^j, o(t)) + h^i(t^{-i})
\end{align*}
\]

where \( h^i \) is an arbitrary function of \( t^{-i} \).

The proof can be done in analogy to the non-weighted case.
A first example: 3 agents are bidding for 2 cars

3 agents are bidding for 2 cars

3 Bidders
1. 400 € / 1 Car
2. 250 € / 1 Car
3. 600 € / 2 Cars

2 Cars

Agents’ types: \( t^1 = 400, \ t^2 = 250, \ t^3 = 600 \)
Feasible outputs: \( F = \{ \{1, 2\}, \{3\}\} \)
The agents’ valuations

Our agents are able to resell their cars for the money indicated by their type (e.g. cars are less worth in Moscow than in Berlin). Therefore their valuations are:

\[ v^i = \begin{cases} t^i & \text{if agent } i \text{ is a winner} \\ 0 & \text{else} \end{cases} \]

Now, what could a simple auction mechanism look like?
A simple auction mechanism

"Sell the cars to the most bidding party!"

What will happen? Since our agents want to make some profit they won’t bid their true valuation, but a bit less (let’s say -100). Agents’ declarations: \( d^1 = 300, d^2 = 150, d^3 = 500 \)

Thus our mechanism is no truthful implementation! The cars will be sold to agent 3 for 500 bucks.

Can we do any smarter?
A first example: 3 agents are bidding for 2 cars

A VGC auction mechanism

"Sell the cars to the most bidding party and take only the opportunity costs that the winner’s presence introduces to all the other bidders!"
(OPportunity cost for agent equals the second highest bid minus that part of the highest bid which doesn’t include agent i itself.) Since Opportunity costs are lower than the highest bid the agents will even make profit by bidding their real values:

**Agents’ declarations**: \( d^1 = 400 \quad d^2 = 250 \quad d^3 = 600 \)

The cars will be sold to agent 1 and agent 2 who bid 650 bucks together. Nevertheless the **agents’ real payments** \( q^i \) are lower:

\[ q^1 = q^3 - q^2 = 350 \quad q^2 = q^3 - q^1 = 200 \quad q^3 = 0 \]
Why does our mechanism belong to the VGC family?

- \( g(t, o) = \sum_{i=1}^{n} v^i(t^i, o) \)
- \( o(t) \in \arg\max_o g(t, o) \)
- \( p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) - (-h^i(t^{-i})) = -q^i(t) \)

where \(-h^i(t^-i)\) is the second highest bid.

Note that \( p^i \) is the "negative" payment the auctioneer hands to the agents and \( q^i = -p^i \) is the "real" payment our agents have to pay to the auctioneer.
A first example: 3 agents are bidding for 2 cars

**Drawbacks of the VGC algorithm**

Agents’ types: \( t^1 = 400, t^2 = 400, t^3 = 300 \)

Feasible outputs: \( F = \{\{1, 2\}, \{3\}\} \)
Results of the VGC auction mechanism

We use the same VGC auction mechanism as before. Therefore the agents bid truthfully:

**Agents' declarations**: \( d^1 = 400 \quad d^2 = 400 \quad d^3 = 300 \)

The cars will be sold to agent 1 and agent 2 who bid 800 bucks together. But note what will happen now. What is the payment our agents have to pay to the auctioneer?

\[
q^1 = q^3 - q^2 = -100 \\
q^2 = q^3 - q^1 = -100 \\
q^3 = 0
\]

In this case our auctioneer will give the 2 cars away with some extra 200 bucks. The mechanism is truthful, but our auctioneer won’t be happy ...
The (not so famous) WAD-Problem

The "Washing-up and drying the dishes"-Problem knows 3 agents which represent connections in a bi-connected graph:

- Knut is able to wash the dishes in $t^K = 5$ min
- Ilse is able to dry the dishes in $t^I = 5$ min
- The Dishwasher can wash and dry the dishes in $t^D = 40$ min
The agents valuations

We define the valuation of an agent $x$ as:

$$v^x = \begin{cases} -t^x & \text{if } x \text{ is part of the chosen washing path} \\ 0 & \text{else} \end{cases}$$

This means that each agent’s valuation is 0 if it is not chosen or the negative time it needs to finish its job, if it is chosen. All agents try to maximize their utility. The utility function of agent $x$ is

$$u^x = v^x + p^x$$

with $p^x$ being the payment defined by our mechanism.
A second example: The WAD-Problem

There exist 2 feasible paths for the WAD-Problem

To implement a truthful mechanism we pay some pocket money ($p^K$, $p^I$) to Knut and Ilse and spend some time ($p^D$) for cleaning the dishwasher. Since time is money we say that we pay out Knut, Ilse and the Dishwasher in time units according to the following rules...
The mechanism’s payment function

We define the **mechanism’s payment function** for agent $x$ as

$$p^x = \begin{cases} 
  d_{G|t^x=\infty} - d_{G|t^x=0} & \text{if } x \text{ is part of the chosen washing path} \\
  0 & \text{else}
\end{cases}$$

where $d_{G|t^x=\infty}$ is the duration of the shortest washing path which doesn’t include agent $x$ and $d_{G|t^x=0}$ stands for the length of the shortest washing path when agent’s $x$ time is assumed to be 0.

Note that $d_{G|t^x=\infty}$ and $d_{G|t^x=0}$ refer to the reported times ($t^x_r$). The real times ($t^x$) are private knowledge to the agents.
A second example: The WAD-Problem

Relation to the VGC mechanism

Our mechanism is indeed a VGC mechanism:

- Our mechanism minimizes the path length (time needed) which corresponds to maximizing the objective function $g(o, t) = \sum_i v^i(t^i, o)$ since $v^i = -t^i$ for all agents belong to the path.

- Our mechanism belongs to the VGC-family since identifying $h^i(t^{-i})$ with $d_G|t^x=\infty$ and $\sum_{j \neq i} v^j(t^j, o(t))$ with $d_G|t^x=0$ results in

$$p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) + h^i(t^{-i}) = d_G|t^x=\infty - d_G|t^x=0$$
Example 1

Assumption: Each agent reports its true time.

Reported times: $t_r^D = 40, t_r^K = 5, t_r^I = 5$

$t_r^K + t_r^I = 10 < t_r^D = 40$
⇒ output $o = A \rightarrow B \rightarrow C$.
(Choose Knut and Ilse for washing the dishes)

⇒ Valuation: $v^D = 0, v^K = -5, v^I = -5$
⇒ Payment: $p^D = 0, p^K = 35, p^I = 35$
⇒ Utility: $u^D = 0, u^K = 30, u^I = 30$

⇒ Maximal utility!
Example 2

Assumption: Knut is cheating.

Reported times: \( t_r^D = 40, t_r^K = 36, t_r^I = 5 \)

\( t_r^K + t_r^I = 41 > t_r^D = 40 \)

\[ \Rightarrow \text{output } o = A \rightarrow C. \]

(Choose Dishwasher for washing)

\[ \Rightarrow \text{Valuation: } v^D = -40 \quad v^K = 0 \quad v^I = 0 \]

\[ \Rightarrow \text{Payment: } p^D = 41 \quad p^K = 0 \quad p^I = 0 \]

\[ \Rightarrow \text{Utility: } u^D = 1 \quad u^K = 0 \quad u^I = 0 \]

\[ \Rightarrow \text{Knut won’t do that!} \]
Example 3

Assumption: The Dishwasher is cheating.

Reported times: $t_r^D = 9$, $t_r^K = 5$, $t_r^I = 5$

$t_r^K + t_r^I = 10 > t_r^D = 9 \Rightarrow$ output $o = A \rightarrow C$. (Choose Dishwasher for washing)

$\Rightarrow$ Valuation: $v^D = -40 \quad v^K = 0 \quad v^I = 0$

$\Rightarrow$ Payment: $p^D = 10 \quad p^K = 0 \quad p^I = 0$

$\Rightarrow$ Utility: $u^D = -30 \quad u^K = 0 \quad u^I = 0$

$\Rightarrow$ The Dishwasher won’t do that!
The Task Allocation Problem

- Allocate $k$ tasks to $n$ agents
- $t^i_j$ is the time needed by agent $i$ to perform task $j$
- Feasible outputs are all partitions $x = x^1...x^n$ where $x^i$ is the set of tasks allocated to agent $i$
- Agent $i$’s valuation is $v^i(x, t^i) = - \sum_{j \in x^i} t^i_j$
- Our goal is to minimize the make-span $g(x, t) = \max_i \sum_{j \in x^i} t^i_j$

We denote a direct revelation mechanism for the task scheduling problem by $m = (x(t), p(t))$, where $x$ is the allocation algorithm (former output) and $p$ stands for the payment.
An Upper Bound

Definition (The MinWork Mechanism)

The idea is to minimize the total work done. This is no very good solution since our agents are able to work in parallel.

- Each task is allocated to the agent who is capable of doing it in a minimal amount of time.
- Each agent is given payment equal to the time of the second best agent for every task: $p^i(t) = \sum_{j \in x^i(t)} \min_{i' \neq i} t^j_{i'}$

Theorem (MinWork)

MinWork is a truthful $n$-approximation mechanism.
An Upper Bound

Proof.

1 MinWork is an n-approximation:
   In the worst case one agent $\hat{i}$ offers the best execution times for all tasks: $\exists \hat{i} \forall i \forall j : t_{\hat{i}}^j \leq t_i^j$. Thus $g(x(t), t) \leq \sum_{j=1}^{k} \min_{i} t_{i}^j$ holds for MinWork.

On the other hand an optimal algorithm can never be faster than doing all $k$ jobs in minimal execution times on all $n$ machines in parallel. Therefore $g(\text{opt}(t), t) \geq \frac{1}{n} \sum_{j=1}^{k} \min_{i} t_{i}^j$ holds for an optimal allocation $\text{opt}(t)$.

Togethe we get: $g(x(t), t) \leq n \times g(\text{opt}(t), t)$. 

An Upper Bound

Proof.

2 MinWork belongs to the VGC family (⇒ is truthful)

The output maximizes \( \sum_{i=1}^{n} v^i(t^i, x) = - \sum_{i=1}^{n} \sum_{j \in x^i} t^i_j \).

On the other hand our payment function can be split up:

\[
p^i(t) = \sum_{j \in x^i(t)} \min_{i' \neq i} t^i_{j'}
= \sum_{j=1}^{k} \min_{i' \neq i} t^i_{j'} - \sum_{j \notin x^i(t)} \min_{i' \neq i} t^i_{j'}
= - \sum_{j \notin x^i(t)} \min_{i' \neq i} t^i_{j'} + \sum_{j=1}^{k} \min_{i' \neq i} t^i_{j'}
= - \sum_{i' \neq i} \sum_{j \in x^{i'}} t^i_{j'} + h^{-i} = \sum_{i' \neq i} \left( - \sum_{j \in x^{i'}} t^i_{j'} \right) + h^{-i}
= \sum_{i' \neq i} v^{i'}(t^i', x) + h^{-i}
\]
A MinWork example

We have 2 agents (Homer, Marge) and 2 tasks (eating, iron). Let's assume our agents have reported their types truthfully. Our mechanism wants to reduce the overall time. We use the MinWork mechanism described above.
An Upper Bound

A MinWork example

Since Homer (agent 1) offers the shortest time for performing both tasks the MinWork mechanism assigns both task to it. Thus $g(x(t), t) = t_1^1 + t_2^1 = 10$. It would be better to handle task 2 to Marge (agent 2): $g(opt(t), t) = \max(t_1^1, t_2^2) = 6$. Alltogether we get a c-approximation with $c = \frac{g(x(t), t)}{g(opt(t), t)} = \frac{10}{6} \approx 1.67 \leq 2$. 
A Lower Bound

We now aim at giving a lower bound for mechanisms that implement the task scheduling problem.

- Due to the **revelation principle** ("if there exists a mechanism that implements a problem with dominant strategies there exists a truthful implementation, too!") it suffices to prove the lower bound for truthful implementations.

- Therefore our mechanism $m$ is always assumed to be truthful!
A Lower Bound

Lemma (Independence)
\[ \forall i, t_1, t_2 : t_1^i = t_2^i \land x^i(t_1) = x^i(t_2) \Rightarrow p^i(t_1) = p^i(t_2) \]

Proof.
Assume \( p^i(t_1) < p^i(t_2) \)! If agent \( i \)'s type is \( t_1^i \) cheating by declaring type \( t_2^i \) would be better \( \Rightarrow \) Contradiction to truthfulness. \( \square \)
A Lower bound

Definition (Price offered for a set of task)

Let $t$ be a type vector and let $X$ be a set of tasks. Then:

$$p^i(X, t^{-i}) = \begin{cases} p^i(\tilde{t}^i, t^{-i}) & \text{if there exists } \tilde{t}^i \text{ s.t. } x^i(\tilde{t}^i, t^{-i}) = X \\ 0 & \text{otherwise} \end{cases}$$

Lemma (Maximization)

For each type vector $t$ and agent $i$ there holds:

$$x^i(t) \in \arg\max_{X \subseteq \{1, \ldots, k\}} (p^i(X, t^{-i}) - t^i(X))$$

Agent $i$'s utility

(The mechanism has to maximize the agent’s benefit. Otherwise the agent will cheat in order to reach maximal benefit itself.)
Basic Lower Bound

We need some preparation in order to prove our main theorem:

**Lemma ("There exist no better allocations like the best one")**

Let $t$ be a type vector and let $X$ be $x^i(t)$. For each set $D \neq X$ of tasks the following inequalities hold:

- $D \subset X \Rightarrow \Delta^i(D, X \setminus D) \geq t^i(X \setminus D)$
- $D \supset X \Rightarrow \Delta^i(X, D \setminus X) \leq t^i(D \setminus X)$
- else $\Rightarrow \Delta^i(L, X \setminus L) - t^i(X \setminus L) \geq \Delta^i(L, D \setminus L) - t^i(D \setminus L)$

(with $L = D \cap X$)

where $\Delta^i(A, B)$ is the price difference $p^i(A \cup B, t^{-i}) - p^i(A, t^{-i})$.

If a set $Y$ of tasks satisfies these inequalities sharply for all $D$'s, then $Y = X = x^i(t)$ ("$X$ is the unique set maximizing $i$’s utility")
Basic Lower Bound

Definition (A type modification)

Let $t$ be a type vector, $i$ an agent, $X$ a set of tasks and let $\alpha > 0$ be a real number. We now denote $\hat{t} = t(X \rightarrow \alpha)$ the type which we obtain by the following case selection

$$\hat{t}_{j}^{'i'} = \begin{cases} 
\alpha & \text{if } i' = i \text{ and } j \in X \\
 t_{j}^{'i'} & \text{otherwise}
\end{cases}$$

Example: Let’s have 2 agents, 2 tasks. Let $\hat{t}$ be $t(\{1, 2\} \rightarrow 0.5)$.

$$t_{1}^{1} = 1.5 \quad t_{1}^{2} = 2.5 \quad t_{2}^{1} = 0.0 \quad t_{2}^{2} = 1.5$$

$$\Downarrow$$

$$\hat{t}_{1}^{1} = 1.5 \quad \hat{t}_{1}^{2} = 2.5 \quad \hat{t}_{2}^{1} = 0.5 \quad \hat{t}_{2}^{2} = 0.5$$
For the following slides, let us assume:

\( n = 2, k \geq 3, \forall i, j : t^i_j = 1 \) and \( |x^1(t)| \leq |x^2(t)| \).

**Lemma (An allocation invariant type modification)**

Let \( 0 < \epsilon < 1, \hat{t} = t(x^1 \xrightarrow{1} \epsilon, x^2 \xrightarrow{1} 1 + \epsilon) \Rightarrow x^1(\hat{t}) = x(t) \)

**Proof.**

Since \( n = 2 \), it is enough to show that \( x^1(\hat{t}) = x^1(t) \). Agent 2’s type is invariant under transformation \( t \rightarrow \hat{t} \). Thus the price offered to agent 1 remains the same. Our transformation therefore only changes the value (time needed). \( x^1(t) \) fulfills the inequalities of the ”best allocations” lemma above. Type transformation to \( \hat{t} \) makes the inequalities strict since \( 0 < \epsilon < 1 \). Therefore the allocation remains the same: \( x^1(\hat{t}) = x(t) \) \( \square \)
Basic Lower Bound

Now we are able to prove **this section’s main theorem:**

**Theorem (Basic Lower Bound)**

*There does not exist a mechanism that implements a* \( c \)-approximation for the task scheduling problem for any \( c < 2 \).

**Proof.**

**Case 1:** \( |x^2(t)| \% 2 = 0 \)

Since \( |x^1(t)| \leq |x^2(t)| \) and \( t \rightarrow \hat{t} \) does not change the allocation there is \( g(x(\hat{t}), \hat{t}) = |x^2(\hat{t})| = |x^2(t)| \). On the other hand an optimal algorithm puts half of agent 2’s load onto agent 1. Therefore agent 1 is now the limiting factor for \( g() \) and we get \( g(\text{opt}(\hat{t}), \hat{t}) \leq \frac{1}{2}|x^2| + k\epsilon \). **Taking it all together we get:**

\[
2 \times (g(\text{opt}) - k\epsilon) \leq g(x(\hat{t}), \hat{t}).
\]
Basic Lower Bound

Proof.

**Case 2:** $|x^2(t)| \% 2 = 1$

Since $k \geq 3$ and $|x^1(t)| \leq |x^2(t)|$ it must be that $|x^2(t)| \geq 3$

Choosing $j \in x^2(t)$ arbitrarily and considering the new type $\hat{t} = \hat{t}({j} \rightarrow \epsilon)$ now still yields the same allocation. Now $g(x(\hat{t}), \hat{t})$ becomes $|x^2| - 1 + \epsilon$. Again an optimal algorithm would ease agent 2’s burden by putting half of the heavy-weight tasks on agent 1 making it the limiting factor: $g(opt(\hat{t}), \hat{t}) \leq \frac{|x^2| - 1}{2} + k\epsilon$.

Taking it all together we get:

$2 \times (g(opt(\hat{t}), \hat{t}) - k\epsilon) + \epsilon \leq |x^2| - 1 + \epsilon = g(x(\hat{t}), \hat{t})$
Resurrecting the MinWork mechanism

Let us think back to the MinWork mechanism now! MinWork is truthful and an \( n \)-approximation to the task scheduling problem. Thus, in the case of 2 agents we have found a 2-approximation mechanism.

The previous lemma has learned us that there exists no mechanism that implements a \( c \)-approximation for the task scheduling problem for any \( c < 2! \)

\[ \Rightarrow \text{(Stupid) MinWork is optimal for 2 agents!} \]
Conjecture

- Nisan and Ronan therefore speculate that there does not exist a mechanism that implements a $c$-approximation for any $c < n$.
- They are not able to prove this conjecture, but they show that it is correct for two cases: additive and local mechanisms.
Thank you for your attention

To be continued ⊞

Any questions?