The impact of cooperation on new high performance computing platforms

Daniel de Angelis Cordeiro

Advisor: Denis Trystram
MOAIS – LIG/INRIA – Université de Grenoble

February 9, 2012

New challenges from e-Science

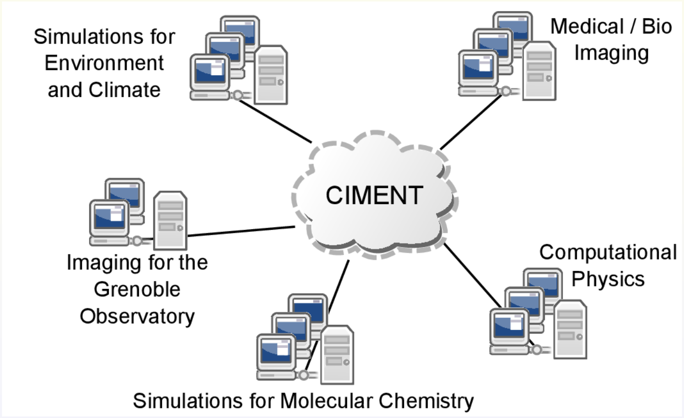
Unity makes strength...

The scientific community has today the unprecedented ability to combine different computational resources into a powerful distributed system capable of analyzing massive data sets.

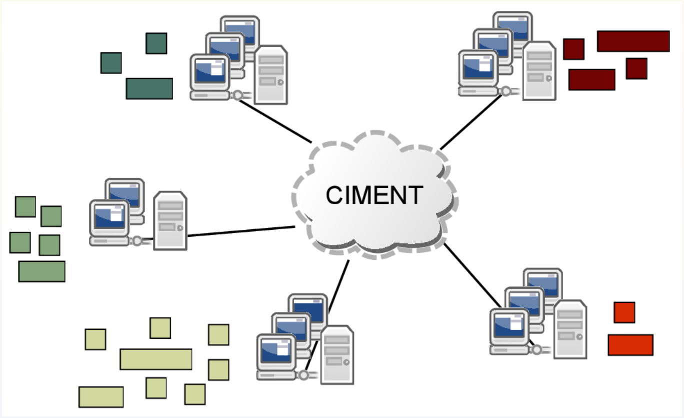
... and the scientific community knows it!

- climate and earth system research;
- genomics and proteomics;
- astrophysics;
- theoretical chemistry;
- physics and high-energy physics.

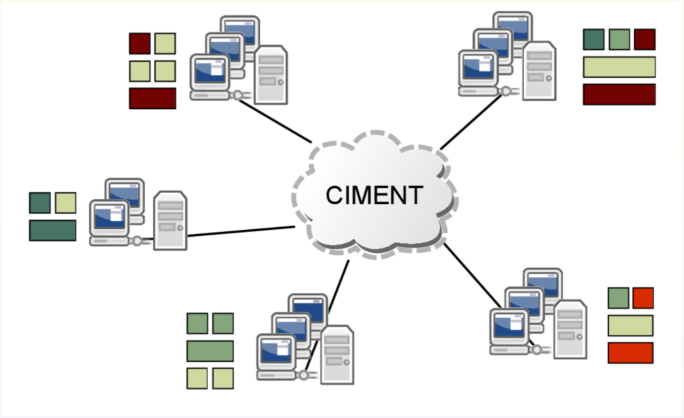
An e-Science grid platform in Grenoble


Simulations for Environment and Climate
Medical / Bio Imaging
CIMENT
Imaging for the Grenoble Observatory
Computational Physics
Simulations for Molecular Chemistry

An e-Science grid platform in Grenoble



An e-Science grid platform in Grenoble



In CIMENT, each participant has a particular objective

Molecular Chemistry

Chemists may be interested in having the results of their simulations as fast as possible.

Objective: to minimize the makespan

Medical / Bio Imaging

Physicians may be interested in delivering results of medical imaging tests, minimizing the average completion time for all users.

Objective: to minimize the average completion time

Question:

How to incentive such different participants to cooperate and share their resources?

In CIMENT, each participant has a particular objective

Molecular Chemistry

Chemists may be interested in having the results of their simulations as fast as possible.

Objective: to minimize the makespan

Medical / Bio Imaging

Physicians may be interested in delivering results of medical imaging tests, minimizing the average completion time for all users.

Objective: to minimize the average completion time

Question:

How to incentive such different participants to cooperate and share their resources?

In CIMENT, each participant has a particular objective

Molecular Chemistry

Chemists may be interested in having the results of their simulations as fast as possible.

Objective: to minimize the makespan

Medical / Bio Imaging

Physicians may be interested in delivering results of medical imaging tests, minimizing the average completion time for all users.

Objective: to minimize the average completion time

Question:

How to incentive such different participants to cooperate and share their resources?

In CIMENT, each participant has a particular objective

Molecular Chemistry

Chemists may be interested in having the results of their simulations as fast as possible.
Objective: to minimize the makespan

Medical / Bio Imaging

Physicians may be interested in delivering results of medical imaging tests, minimizing the average completion time for all users.
Objective: to minimize the average completion time

Question:
How to incentive such different participants to cooperate and share their resources?

In CIMENT, each participant has a particular objective

Molecular Chemistry

Chemists may be interested in having the results of their simulations as fast as possible.
Objective: to minimize the makespan

Medical / Bio Imaging

Physicians may be interested in delivering results of medical imaging tests, minimizing the average completion time for all users.
Objective: to minimize the average completion time

Question:
How to incentive such different participants to cooperate and share their resources?

In CIMENT, each participant has a particular objective

Molecular Chemistry

Chemists may be interested in having the results of their simulations as fast as possible.

Objective: to minimize the makespan

Medical / Bio Imaging

Physicians may be interested in delivering results of medical imaging tests, minimizing the average completion time for all users.

Objective: to minimize the average completion time

Question:

How to incentive such different participants to cooperate and share their resources?

In this thesis

We study four different facets of the rules that govern how different participants engage in cooperation. We show how to use scheduling algorithms to ensure the efficiency of the platform when cooperation takes place between:

- organizations that cannot be completely trusted;
- organizations willing to sacrifice their performance in pro of the society;
- organizations with freedom to choose where to schedule their own jobs;
- users sharing a common set of resources (inside an organization).

Outline

Outline

How to promote cooperation?

Locally

The performance obtained by an organization should be at least as good as the one obtained using only its own machines.

Globally

The utilization of all machines should be maximized. In this context, it means to minimize the global makespan.

Multi-organization scheduling problem

This problem — introduced by Pascual *et al*. in 2007 — is known as the Multi-organization scheduling problem and is denoted by MOSP($C_{\max}$) or MOSP($\sum C_i$).

How to promote cooperation?

Locally

The performance obtained by an organization should be at least as good as the one obtained using only its own machines.

Globally

The utilization of all machines should be maximized. In this context, it means to minimize the global makespan.

Multi-organization scheduling problem

This problem — introduced by Pascual *et al*. in 2007 — is known as the Multi-organization scheduling problem and is denoted by MOSP($C_{\max}$) or MOSP($\sum C_i$).

How to promote cooperation?

Locally

The performance obtained by an organization should be at least as good as the one obtained using only its own machines.

Globally

The utilization of all machines should be maximized. In this context, it means to minimize the global makespan.

Multi-organization scheduling problem

This problem — introduced by Pascual *et al*. in 2007 — is known as the Multi-organization scheduling problem and is denoted by MOSP($C_{\max}$) or MOSP($\sum C_i$).

The Multi-Organization Scheduling Problem

Global objective

The goal is to minimize the global $C_{\max}$ of the entire platform.

Local restrictions

Locally, the scheduler is not allowed to impose any performance degradation to an organization as a side effect of the minimization of the global objective.

Best known result so far

[Dutot *et al.*, 2011] presented a tight 3-approximation algorithm for parallel jobs.

We focus this work on workloads of independent sequential jobs.

The Multi-Organization Scheduling Problem

Global objective

The goal is to minimize the global $C_{\max}$ of the entire platform.

Local restrictions

Locally, the scheduler is not allowed to impose any performance degradation to an organization as a side effect of the minimization of the global objective.

Best known result so far

[Dutot *et al.*, 2011] presented a tight 3-approximation algorithm for parallel jobs.

We focus this work on workloads of independent sequential jobs.

The Multi-Organization Scheduling Problem

Global objective

The goal is to minimize the global $C_{\max}$ of the entire platform.

Local restrictions

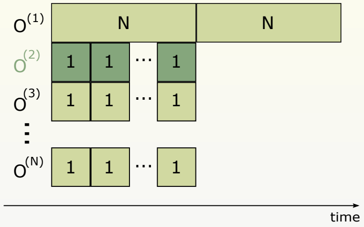
Locally, the scheduler is not allowed to impose any performance degradation to an organization as a side effect of the minimization of the global objective.

Best known result so far

[Dutot *et al.*, 2011] presented a tight 3-approximation algorithm for parallel jobs.

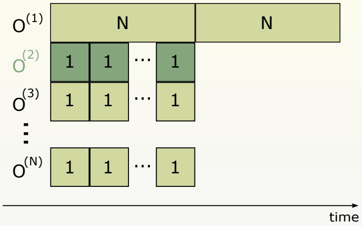
We focus this work on workloads of independent sequential jobs.

Local constraints

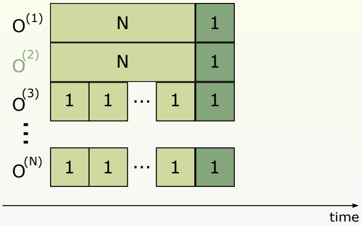


(a) Initial instance

Local constraints



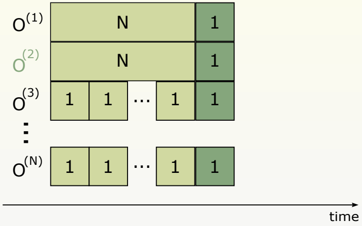
(a) Initial instance



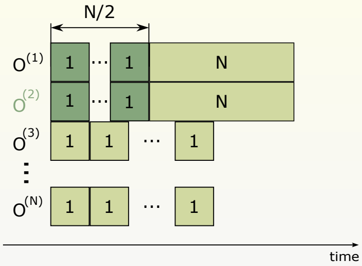
(b) Global optimum without constraints

Scheduling (b) penalizes organization $O^{(2)}$ in order to obtain the optimal makespan.

Local constraints

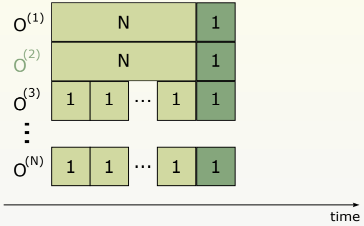


(b) Global optimum without constraints

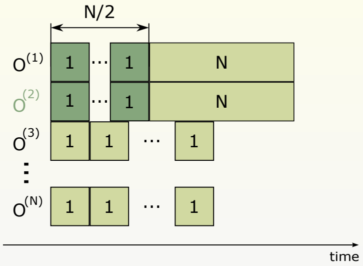


(c) Optimum with MOSP constraints

Local constraints



(b) Global optimum without constraints



(c) Optimum with MOSP constraints

Asymptotic Lower bound

The ratio between the best possible global makespan with ($\frac{3N}{2}$) and without ($N + 1$) the local constraints is asymptotically equal to $\frac{3}{2}$.

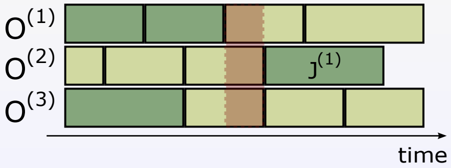
Selfishness

Selfish organizations

A selfish organization can change the schedule devised by the central scheduler in order to give priority to its own jobs.

An organization could change the schedule if:

- its jobs are not prioritized on its own resources;
- a migrated job can be re-inserted earlier on one of its own machines.



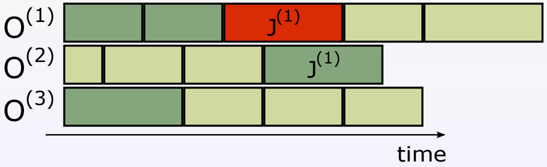
Selfishness

Selfish organizations

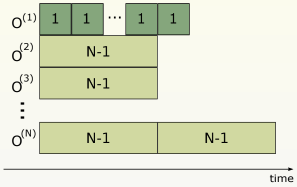
A selfish organization can change the schedule devised by the central scheduler in order to give priority to its own jobs.

An organization could change the schedule if:

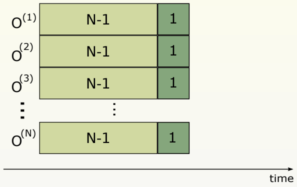
- its jobs are not prioritized on its own resources;
- a migrated job can be re-inserted earlier on one of its own machines.



Inapproximation

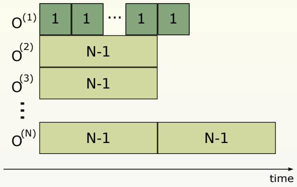


(a) Optimal with selfishness restrictions

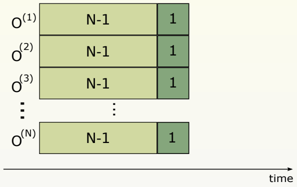


(b) Optimal with local constraints

Inapproximation



(a) Optimal with selfishness restrictions

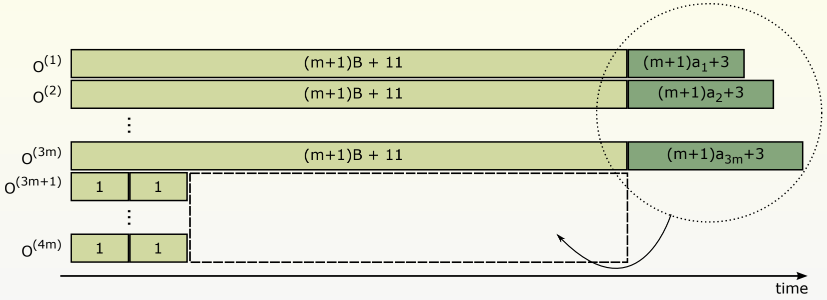


(b) Optimal with local constraints

Lower bound

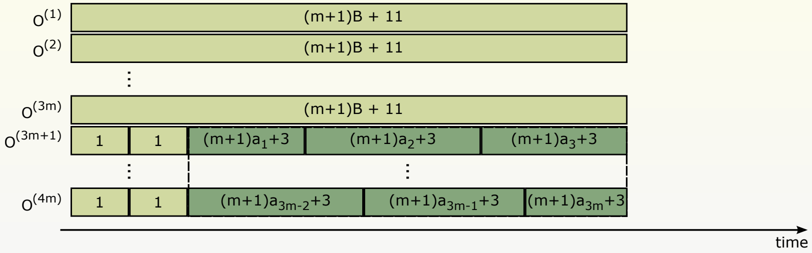
The ratio of the optimal solution with the selfishness restrictions to the optimal solution with MOSP constraints is $2 - \frac{2}{N}$.

Complexity – MOSP($C_{\max}$) and MOSP($\sum C_i$)



Theorem
MOSP($C_{\max}$) *and* MOSP($\sum C_i$) *are strongly NP-complete even if each organization has exactly two jobs. Proof. Reduction from* 3-PARTITION

Complexity – MOSP($C_{\max}$) and MOSP($\sum C_i$)



Idea

Using $4m$ organizations, build $3m$ large jobs that must be scheduled alone, forcing $3m$ smaller jobs (built from a_i) to fit into the m remaining organizations in order to obtain the optimal $C_{\max}$.

Heuristics

We analyzed four heuristics that respect both MOSP and selfishness constraints. All of them work in two phases:

Phase 1 Each organization schedules its jobs locally according to its own local objective;

Phase 2 All organizations cooperatively minimize the global $C_{\max}$: each time a processor becomes idle, then:

- **LPT-LPT** and **SPT-LPT**: the longest job that hasn't started yet is migrated to the idle processor;
- **Less Helped First**: a job from the organization that had less work executed by others is migrated to the idle processor;
- **ILBA**: all jobs from an entire organization (from the less loaded to the most loaded) are rebalanced.

Properties

- The cooperative phase works as a list scheduling, so Graham's classical approximation ratio of $2 - \frac{1}{N}$ holds.
- All migrated jobs are executed earlier:
  - selfishness restriction is always respected;
  - both $C_{\max}$ and $\sum C_i$ are decreased.
- No further enhancements are possible without removing selfishness restrictions.

Experimental analysis

According to the obtained global $C_{\max}$

- For large number of jobs: ILBA and LPT-LPT results are near optimal;
- When the ratio of the number of jobs to the number of machines is low: LPT-LPT performance is better.

Experimental analysis

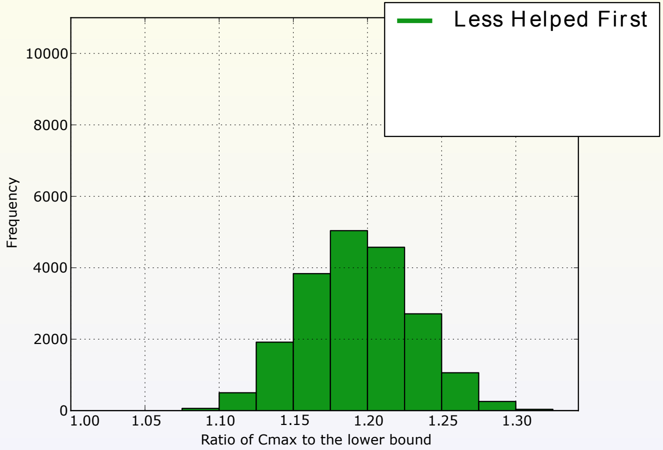


Figure: Frequency of results obtained on 20,000 instances with 20 machines and 100 jobs.

Experimental analysis

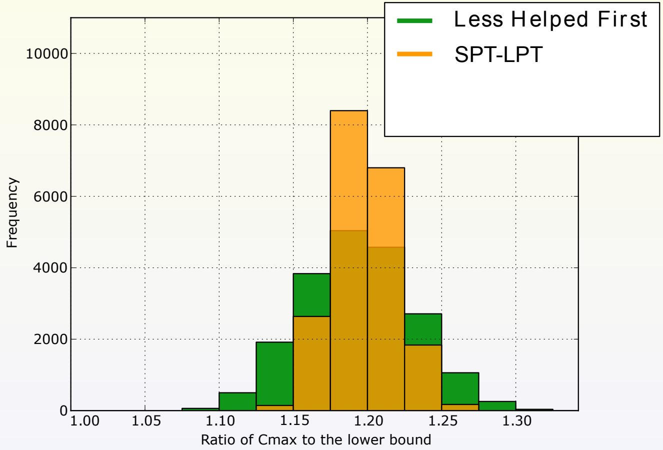


Figure: Frequency of results obtained on 20,000 instances with 20 machines and 100 jobs.

Experimental analysis

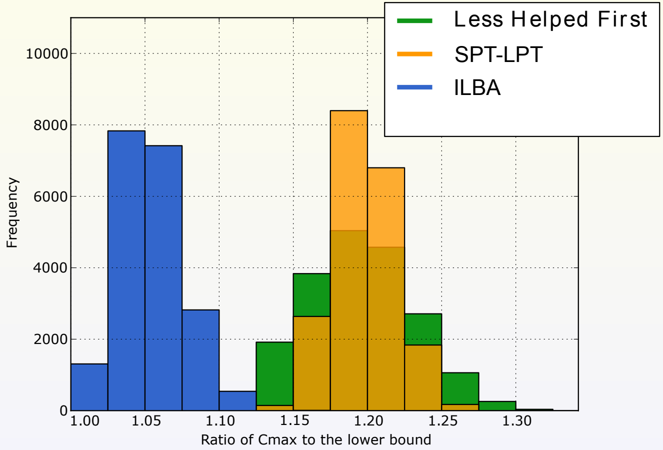


Figure: Frequency of results obtained on 20,000 instances with 20 machines and 100 jobs.

Experimental analysis

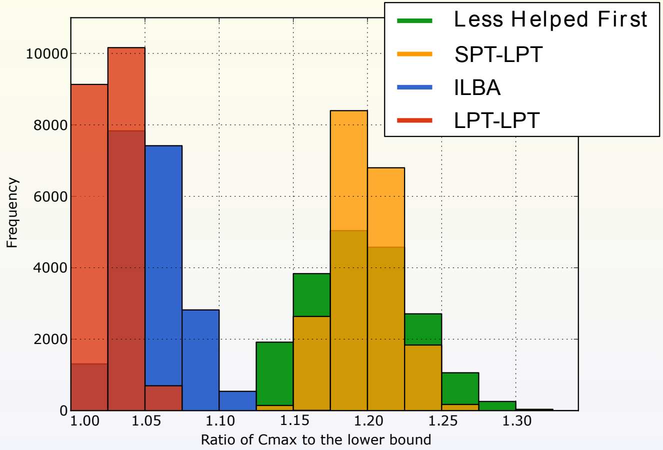
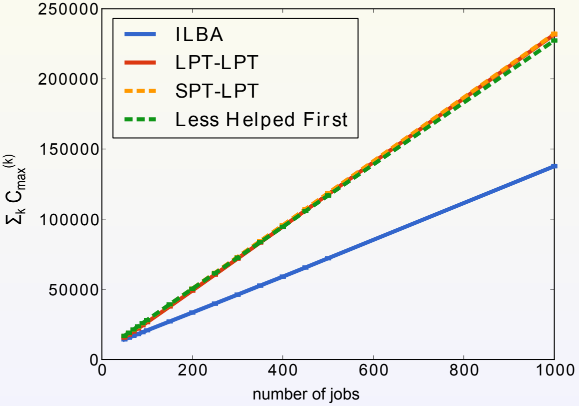


Figure: Frequency of results obtained on 20,000 instances with 20 machines and 100 jobs.

According to the local C_{max}

- ILBA is more aggressive on improving the local C_{max}.



Outline

Local vs. Global

Strict constraints

MOSP's local and selfishness constraints are too strict in practice. They strongly limit the freedom of the scheduler to find a good global $C_{\max}$.

A clear trade-off

There is a correlation between the guarantees that we can provide individually for each organization and the global performance of the platform.

Question

How much can we improve the global $C_{\max}$ of the entire platform if we allow some controlled degradation of the local performance?

Local vs. Global

Strict constraints
MOSP's local and selfishness constraints are too strict in practice. They strongly limit the freedom of the scheduler to find a good global $C_{\max}$.

A clear trade-off
There is a correlation between the guarantees that we can provide individually for each organization and the global performance of the platform.

Question
How much can we improve the global $C_{\max}$ of the entire platform if we allow some controlled degradation of the local performance?

Local vs. Global

Strict constraints

MOSP's local and selfishness constraints are too strict in practice. They strongly limit the freedom of the scheduler to find a good global $C_{\max}$.

A clear trade-off

There is a correlation between the guarantees that we can provide individually for each organization and the global performance of the platform.

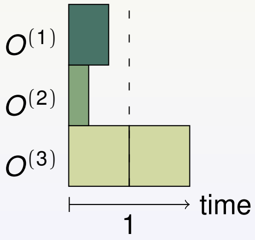
Question

How much can we improve the global $C_{\max}$ of the entire platform if we allow some controlled degradation of the local performance?

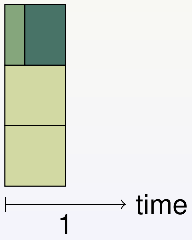
The α-Cooperative Multi-Organization Scheduling Problem

Definition

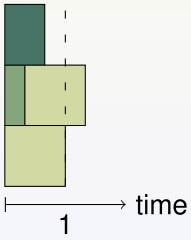
We denote as $(\alpha;\ \beta)$ a schedule that allows the local objective to be degraded by a factor α in order to guarantee a β-approximation for the global $C_{\max}$.



(a) Initial instance



(b) $\left(\alpha = \frac{3}{2};\ \beta = 1\right)$



(c) $\left(\alpha = 1;\ \beta = \frac{4}{3}\right)$

Inapproximability for Family 1 (better global $C_{\max}$)

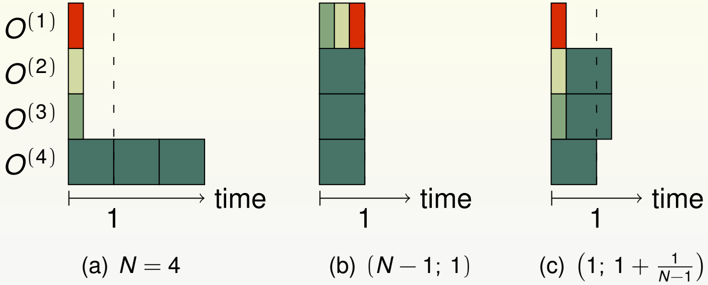


Figure: Family 1 for $N = 4$

In other words:

For $N = 4$, no schedule can have a ratio of $\left(3-\epsilon;\ \frac{4}{3}-\epsilon\right)$.

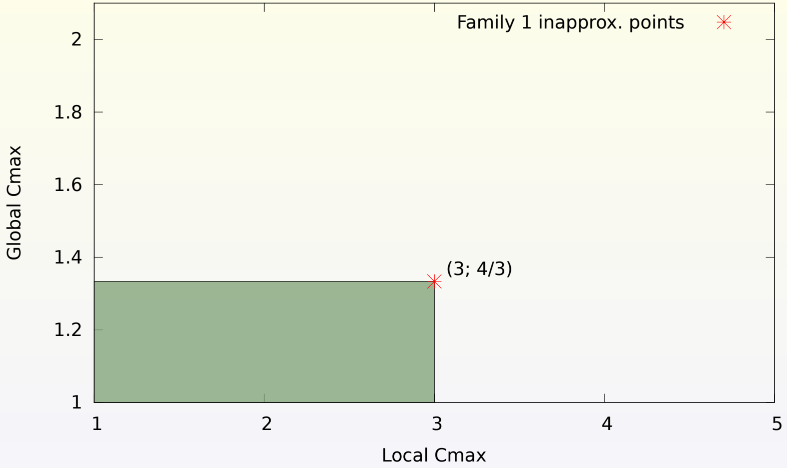


Figure: Inapproximation points given by Family 1 for $N = 4$

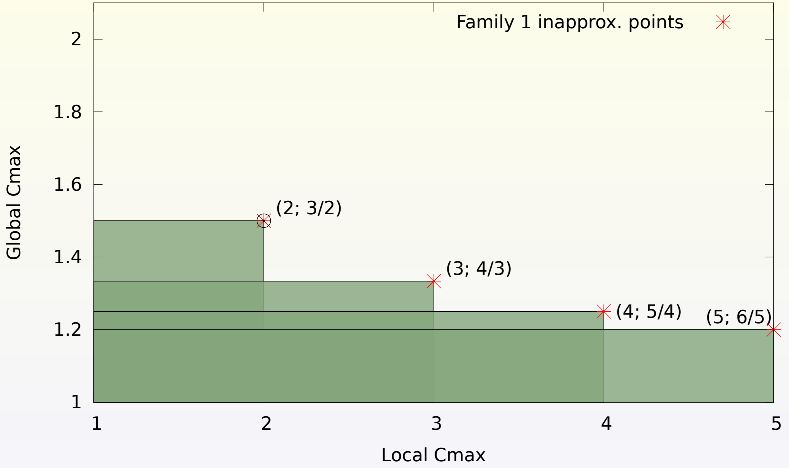


Figure: Inapproximation points given by Family 1 for $N = 3, 4, 5, 6$

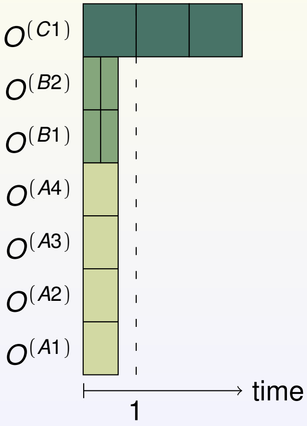
Inapproximability for Family 2 (better local $C_{\max}$)

Let j, k be integers such that $j > 1$ and $k > j - 2$.

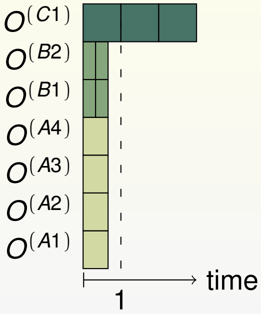
$O^{(A)}$: $(j-1)k$ organizations;
1 job of length $\frac{j-1}{j}$;

$O^{(B)}$: k organizations;
$j - 1$ jobs of length $\frac{1}{j}$;

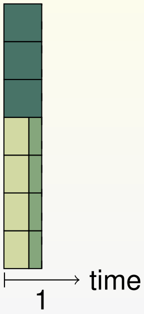
$O^{(C)}$: 1 organization;
$k + 1$ jobs of length 1.



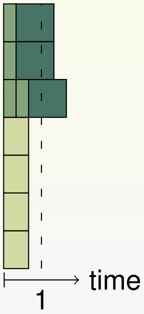
Inapproximability for Family 2 (better local $C_{\max}$)



(a) $(j = 3; k = 2)$

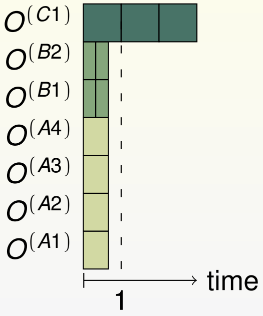


(b) $\left(\frac{j}{j-1};\ 1\right)$

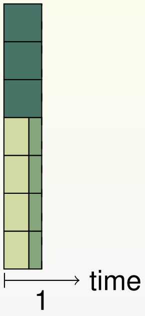


(c) $\left(1;\ 1 + \frac{j-1}{j}\right)$

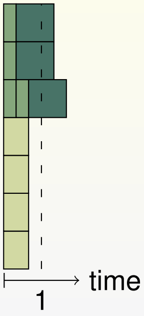
Inapproximability for Family 2 (better local $C_{\max}$)



(a) $(j = 3; k = 2)$



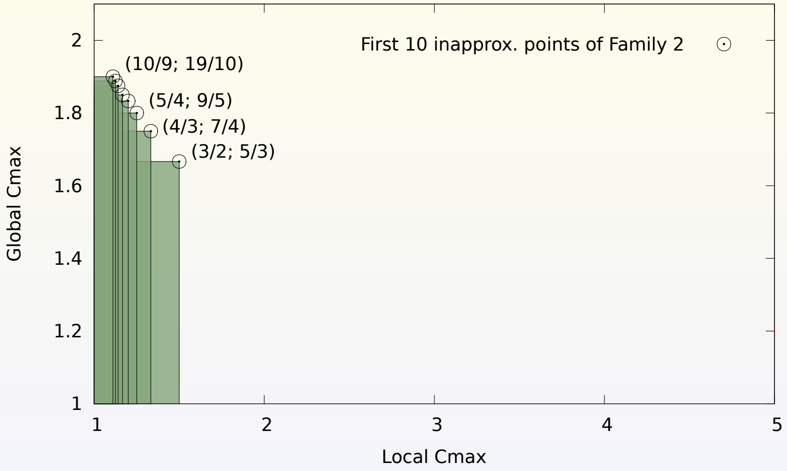
(b) $\left(\frac{j}{j-1};\ 1\right)$

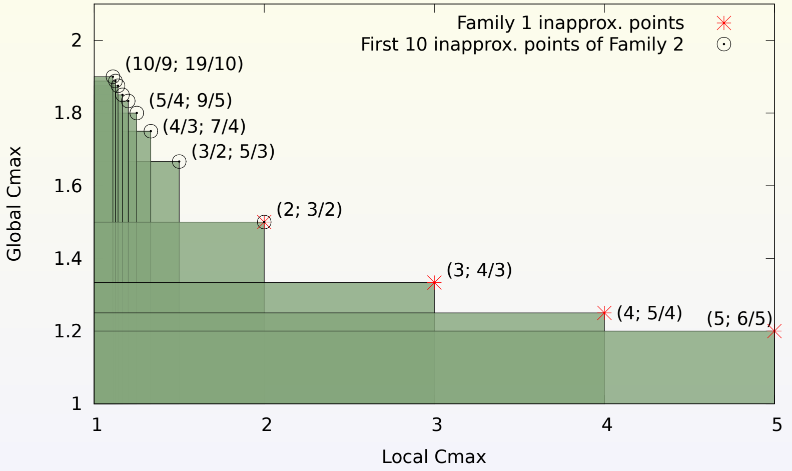


(c) $\left(1;\ 1 + \frac{j-1}{j}\right)$

In this example

No schedule can have a ratio of $\left(\frac{3}{2} - \epsilon;\ \frac{5}{3} - \epsilon\right)$.


First 10 inapprox. points of Family 2 ⊙
(10/9; 19/10)
(5/4; 9/5)
(4/3; 7/4)
(3/2; 5/3)
Global Cmax
2
1.8
1.6
1.4
1.2
1
Local Cmax
1
2
3
4
5


Family 1 inapprox. points
First 10 inapprox. points of Family 2
(10/9; 19/10)
(5/4; 9/5)
(4/3; 7/4)
(3/2; 5/3)
(2; 3/2)
(3; 4/3)
(4; 5/4)
(5; 6/5)
Global Cmax
Local Cmax
2
1.8
1.6
1.4
1.2
1
1
2
3
4
5

We present two algorithms for the problem with guaranteed ratios of $\left(2; \frac{3}{2}\right)$ and $\left(3; \frac{4}{3}\right)$.

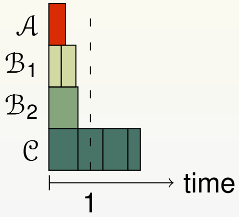


Figure: Classification

Algorithm 1:

- $\mathcal{A} = \{O^{(k)} \mid C_{\max}^{(k)\ local} \leqslant \frac{1}{2}\}$;
- $\mathcal{B}_1 = \{O^{(k)} \mid \frac{1}{2} < C_{\max}^{(k)\ local} \leqslant \frac{3}{4}$ and $\nexists\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{2}\}$;
- $\mathcal{B}_2 = \{O^{(k)} \mid \frac{1}{2} < C_{\max}^{(k)\ local} \leqslant \frac{3}{4}$ and $\exists!\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{2}\}$;
- $\mathcal{C} = \{O^{(k)} \mid C_{\max}^{(k)\ local} > \frac{3}{4}\}$.

We present two algorithms for the problem with guaranteed ratios of $\left(2;\ \frac{3}{2}\right)$ and $\left(3;\ \frac{4}{3}\right)$.

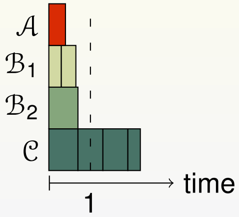


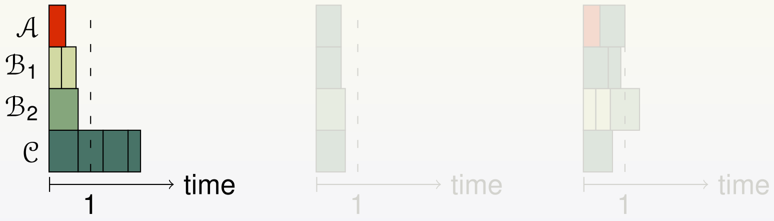
Figure: Classification

Algorithm 2:

- $\mathcal{A} = \{O^{(k)} \mid C_{\max}^{(k)\ local} \leqslant \frac{1}{3}\}$;
- $\mathcal{B}_1 = \{O^{(k)} \mid \frac{1}{3} < C_{\max}^{(k)\ local} \leqslant \frac{4}{9}$ and $\nexists\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{3}\}$;
- $\mathcal{B}_2 = \{O^{(k)} \mid \frac{1}{3} < C_{\max}^{(k)\ local} \leqslant \frac{4}{9}$ and $\exists!\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{3}\}$;
- $\mathcal{C} = \{O^{(k)} \mid C_{\max}^{(k)\ local} > \frac{4}{9}\}$.

We present two algorithms for the problem with guaranteed ratios of $\left(2;\ \frac{3}{2}\right)$ and $\left(3;\ \frac{4}{3}\right)$.

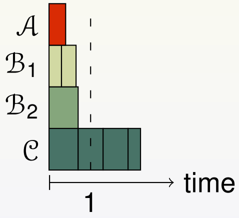
General principle of the algorithms



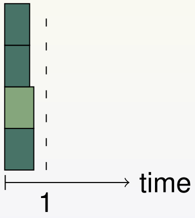
(a) Classification (b) Place large jobs (c) Pairing

We present two algorithms for the problem with guaranteed ratios of $\left(2;\ \frac{3}{2}\right)$ and $\left(3;\ \frac{4}{3}\right)$.

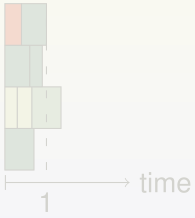
General principle of the algorithms



(a) Classification



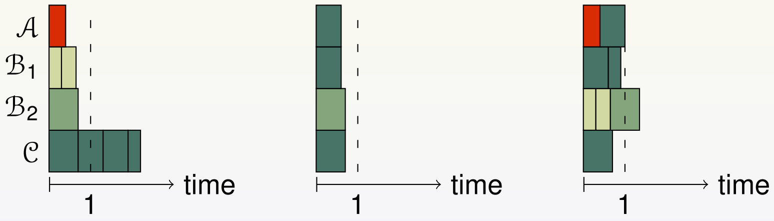
(b) Place large jobs



(c) Pairing

We present two algorithms for the problem with guaranteed ratios of $\left(2;\ \frac{3}{2}\right)$ and $\left(3;\ \frac{4}{3}\right)$.

General principle of the algorithms



(a) Classification (b) Place large jobs (c) Pairing

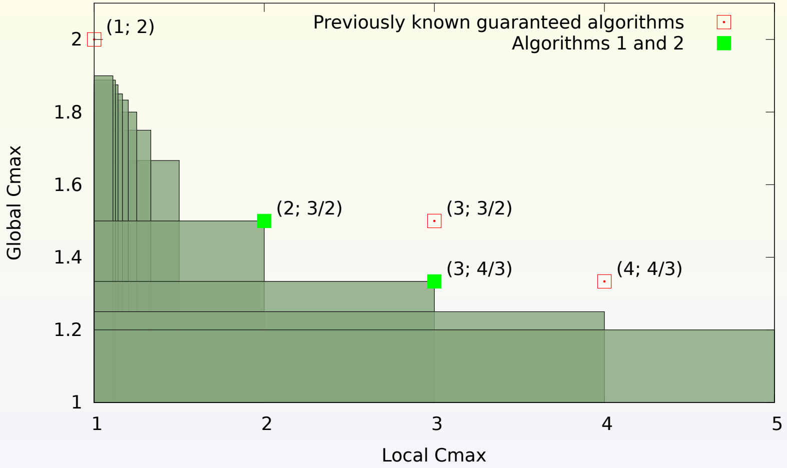


Figure: Summary of the results, including the previously known points $(1;\ 2)$ (MOSP); $(3;\ \frac{3}{2})$ and $(4;\ \frac{4}{3})$ from [Ooshita *et al.*, 2009].

Outline

Individualism of the organizations

Another possible form of incentive to cooperation is to give to each participant more power on the decision-making process.

Game theoretic analysis

Using algorithmic game-theory, we extend the notions of independence and selfishness of each organization. The goal is to study the interactions between the independent organizations as the result of rational selfish players attempting to reach an equilibrium.

Basic model

We study MOSP as a non-cooperative game modeled as follows:

- each agent is responsible for the assignment of jobs belonging to one organization;
- agents are rational and always choose their *best response*;
- the cost function of an agent responsible for $O^{(k)}$ is given by $\text{cost}^{(k)} = C_{\max}^{(k)}$;
- each organization will compute a schedule to the jobs assigned to it using a coordination mechanism.

Nash equilibria

We are interesting in study the global makespan produced when no organization can unilaterally improve its local makespan.

Pure Nash equilibrium

A configuration M is a pure Nash equilibrium if all agent k satisfies the following property: $\forall s \in \mathcal{S}^{(k)}$, $\text{cost}^{(k)}(M) \leqslant \text{cost}^{(k)}(s, M_{-k})$, where M_{-k} is a vector $(S^{(1)}, S^{(2)}, S^{(k-1)}, S^{(k+1)} \ldots, S^{(N)})$.

Pure ϵ-approximate Nash equilibrium

A configuration M is an ϵ-approximate equilibrium if it holds that: $\forall s \in \mathcal{S}^{(k)}$, $\text{cost}^{(k)}(M) \leqslant \epsilon \times \text{cost}^{(k)}(s, M_{-k})$.

Coordination mechanisms

The *coordination mechanism* defines how one organization will schedule and execute locally the jobs that were assigned to it.

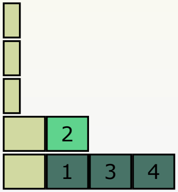
games with priority to jobs
: each organization schedules first its own jobs and then schedules the foreign jobs using some classical scheduling algorithm (LPT, SPT, etc.), assigning a priority calculated separately for each job;

games with priority to organizations
: each organization schedules first its own jobs and then the foreign jobs prioritizing them according to the organization that owns the job.

On game with priorities to jobs

Theorem

MOSP *games defined with a coordination mechanism that assigns priorities to jobs independently of the owner organization do not admit a pure* ϵ*-approximate equilibrium for values of* $\epsilon < 2$.

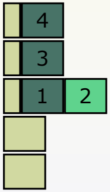


Initial instance

On game with priorities to jobs

Theorem

MOSP *games defined with a coordination mechanism that assigns priorities to jobs independently of the owner organization do not admit a pure* ϵ*-approximate equilibrium for values of* $\epsilon < 2$.

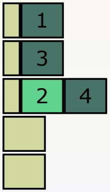


$$C_{max}^{(4)} = 2 + 5\delta$$
$$C_{max}^{(5)} = 1 + 3\delta$$

On game with priorities to jobs

Theorem

MOSP *games defined with a coordination mechanism that assigns priorities to jobs independently of the owner organization do not admit a pure ϵ-approximate equilibrium for values of $\epsilon < 2$.*

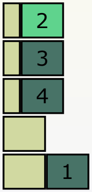


$$C_{\max}^{(4)} = 1 + 3\delta$$
$$C_{\max}^{(5)} = 2 + 5\delta$$

On game with priorities to jobs

Theorem

MOSP *games defined with a coordination mechanism that assigns priorities to jobs independently of the owner organization do not admit a pure* ϵ*-approximate equilibrium for values of* $\epsilon < 2$.

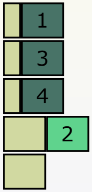


$$\mathbf{C}_{\mathbf{max}}^{(4)} = \mathbf{1} + \mathbf{3}\delta$$
$$\mathbf{C}_{\mathbf{max}}^{(5)} = \mathbf{2} + \mathbf{4}\delta$$

On game with priorities to jobs

Theorem

MOSP *games defined with a coordination mechanism that assigns priorities to jobs independently of the owner organization do not admit a pure ϵ-approximate equilibrium for values of $\epsilon < 2$.*



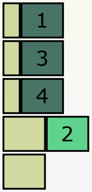
$$C_{max}^{(4)} = 2 + 4\delta$$
$$C_{max}^{(5)} = 1 + 3\delta$$

Loop! No pure ϵ-approximate equilibrium exists unless $\epsilon \geqslant \frac{2+4\delta}{1+3\delta} \Rightarrow \epsilon \geqslant 2$.

On game with priorities to jobs

Theorem

MOSP *games defined with a coordination mechanism that assigns priorities to jobs independently of the owner organization do not admit a pure ϵ-approximate equilibrium for values of $\epsilon < 2$.*



Complexity

Even knowing if a particular instance admits a pure equilibrium is co-NP-hard.

On games with priorities to organizations

Theorem

Given a list scheduling approximation algorithm, the follow algorithm constructs a pure ϵ-approximate Nash equilibrium:

```
1: for all $O^{(k)} \in \{O^{(1)}, O^{(2)}, \ldots, O^{(N)}\}$ do
2:     locally schedule all jobs belonging to $O^{(k)}$ using the list
       scheduling approximation algorithm;
3: end for
4: for all $O^{(k)}$ from the highest to the lowest priority do
5:     unschedule all jobs from $O^{(k)}$;
6:     reschedule the jobs using the approximation algorithm on all
       available processors;
7: end for
```

Price of Anarchy

How inefficient is a system where each selfish player makes its own decisions if compared to an idealized situation, where all players would collaborate selflessly?

Price of anarchy

Given S the set of possible strategies, $E \subseteq S$ the set of all Nash equilibria and $C : S \mapsto \mathbb{R}$ a cost function that measures the inefficiency of the system, the *price of anarchy* (PoA) is defined by:

$$\text{PoA} = \frac{\max_{s \in E} C(s)}{\min_{s \in S} C(s)}$$

Theorem

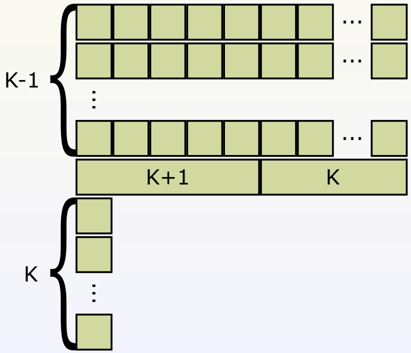
The price of anarchy (PoA) of MOSP *games with priorities given to organizations is lower or equal to* $2 - \frac{1}{N}$, *and this bound is asymptotically tight.*

Sketch of the proof:

Theorem

The price of anarchy (PoA) of MOSP *games with priorities given to organizations is lower or equal to* $2 - \frac{1}{N}$*, and this bound is asymptotically tight.*

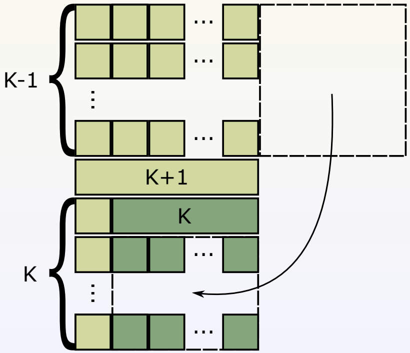
Sketch of the proof:



Theorem

The price of anarchy (PoA) of MOSP *games with priorities given to organizations is lower or equal to* $2 - \frac{1}{N}$, *and this bound is asymptotically tight.*

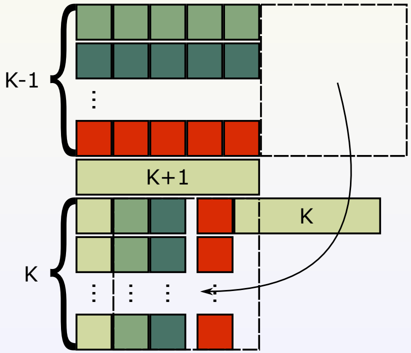
Sketch of the proof:



Theorem

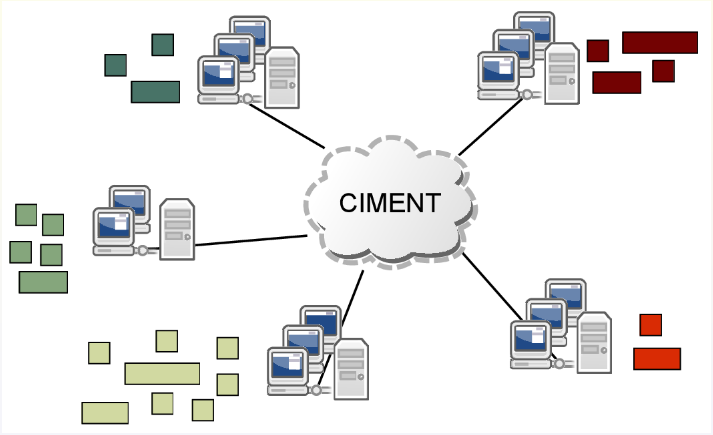
The price of anarchy (PoA) of MOSP *games with priorities given to organizations is lower or equal to* $2 - \frac{1}{N}$*, and this bound is asymptotically tight.*

Sketch of the proof:



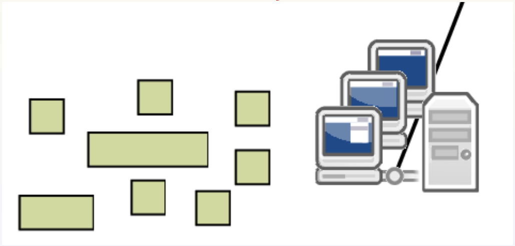
Outline

Shared resources



Shared resources

Users with different objectives



Shared resources

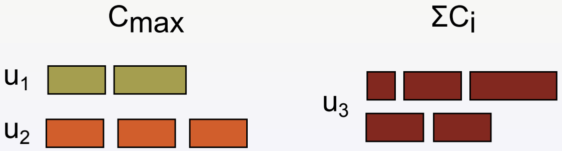
Users with different objectives

Problem

How to guarantee fairness for users with different needs?

In the Multi-Users Scheduling Problem, denoted by $\mathrm{MUSP}(k' : \sum C_i;\ k'' : C_{\max})$, k' users are interested in minimizing the average completion time of their jobs and k'' users are interested in their $C_{\max}$.

Example



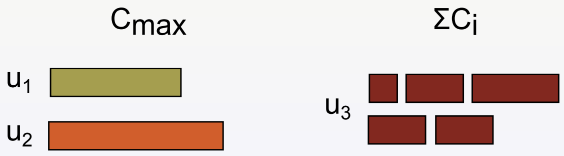
sharing 1 machine

Problem

How to guarantee fairness for users with different needs?

In the Multi-Users Scheduling Problem, denoted by $\text{MUSP}(k' : \sum C_i;\ k'' : C_{\max})$, k' users are interested in minimizing the average completion time of their jobs and k'' users are interested in their $C_{\max}$.

Example



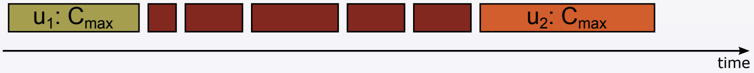
sharing 1 machine

Problem

How to guarantee fairness for users with different needs?

In the Multi-Users Scheduling Problem, denoted by $\text{MUSP}(k' : \sum C_i;\ k'' : C_{\max})$, k' users are interested in minimizing the average completion time of their jobs and k'' users are interested in their $C_{\max}$.

Example



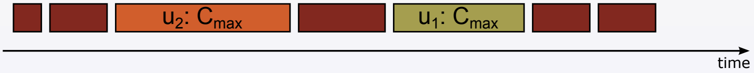
u_1 gets the best C_{max} possible, u_2 the worst

Problem

How to guarantee fairness for users with different needs?

In the Multi-Users Scheduling Problem, denoted by $\text{MUSP}(k' : \sum C_i;\ k'' : C_{\max})$, k' users are interested in minimizing the average completion time of their jobs and k'' users are interested in their $C_{\max}$.

Example



How to mix these objectives?

Novel method for finding fair solutions with good compromises

Clear trade-off between the objectives

How to find solutions not too far from the Pareto set and presenting good fairness?

Our method:

- uses [Papadimitriou and Yannakakis, 2000] decomposition of the solution space on hyperrectangles to search for an ϵ-approximate Pareto set;
- the Pareto set can be exponentially large, we use Pareto approximations for MUSP to find one point by hyperrectangle;
- guided by a fairness function, we refine these hyperrectangles iteratively in order to find solutions with better fairness.

Outline

When cooperation occurs between:

Organizations that cannot be completely trusted

- using classical combinatorial optimization tools, we studied the relations between the global and local objectives of selfish organizations sharing resources and jobs;
- we provided lower bounds, approximation algorithms and an experimental analysis on how much an organization can improve its performance.

When cooperation occurs between:

Organizations willing to sacrifice their performance in pro of the society

- MOSP constraints are too strict in practice, the ratio $(1; 2)$ can be improved by relaxing these constraints;
- this led to the study of a bi-criteria scheduling problem that relates the local degradation and the global performance;
- we presented guaranteed algorithms and a study on the inapproximability of the problem.

When cooperation occurs between:

Organizations with freedom to choose where to schedule their jobs

- we investigated game theoretic approaches for MOSP, expecting to have a decentralized scheduling approach;
- we showed that with the right coordination mechanisms it is possible to have pure Nash equilibria with bounded price-of-anarchy;
- but we feel that game-theory was not the best tool for this because of the centralized nature of its analysis tools.

When cooperation occurs between:

Users sharing a common set of resources

- we studied the problem of scheduling inside each organization, where different users compete for the resources;
- using tools from multi-objective optimization, we started a work on how to use fairness to find solutions with good compromises for the users;
- this study needs further work on new Pareto approximation for MUSP and more experiments on different problems.

Research perspectives

Short-term projects

- Better algorithms for α-MOSP: we are working on an algorithm with guarantee $\left(\frac{3}{2};\ \frac{5}{3}\right)$.
- More Pareto approximations for MUSP, allowing more experimental analyses using the new search method and applications to other problems.

Long-term projects

- Applications for MOSP on more dynamic platforms (like cloud computing).
- Optimization when users' and organizations' interests are considered simultaneously.

Thank you for your attention.

Shared resources

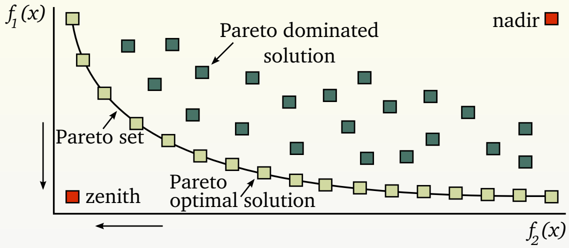
In the MOSP problem:

- each organization contributes with *resources*
- each organization could stop cooperating and execute its jobs by itself

What if the participants must share the same resources?

- how to respect the individual interests of the users?
- how to provide performance guarantees for each user?
- how to achieve fairness in such schedules?

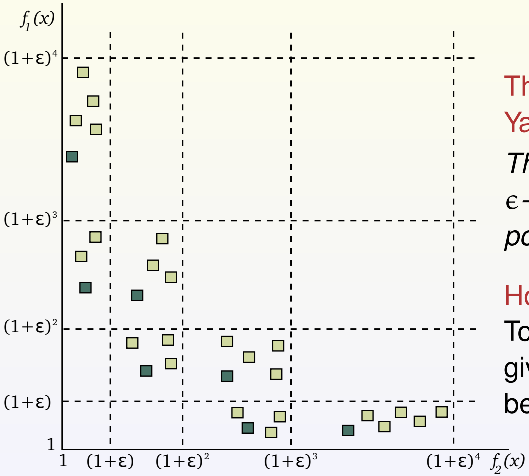
Solutions with good trade-off and fairness



Our goal

Search among solutions not too far from the Pareto set, the ones that optimize a monotonic objective function (like, e.g., fairness functions).

General principle of the search method



Theorem (Papadimitriou and Yannakakis, 2000)

There is always an ϵ-approximate Pareto set of polynomial size.

However

To find a solution inside one given hyperrectangle may not be algorithmically easy.

General principle of the search method

Approximating using thresholds

If one has a <$\rho_1, \rho_2, \ldots, \rho_x, \overline{\rho_{x+1}}, \overline{\rho_{x+2}}, \ldots, \overline{\rho_k}$>-approximation algorithm *A*, then it is possible to compute a $(\rho_1, \ldots, \rho_k)$-approximation of the nadir of the hyperboxes that contain solutions.

The nadir is a $(1+\epsilon)$-approximation of the zenith of the hyperbox

So, *A* returns a $((1+\epsilon)\rho_1, \ldots, (1+\epsilon)\rho_k)$-approximation of the zenith and the set of solutions on all hyperboxes is a $((1+\epsilon)\rho_1, \ldots, (1+\epsilon)\rho_k)$-approximation of the Pareto set.

Application to the MUSP problem

In the Multi-Users Scheduling Problem, denoted by MUSP($k' : \sum C_i;\ k'' : C_{\max}$), k' users are interested in minimizing the average completion time of their jobs and k'' users interested in their $C_{\max}$.

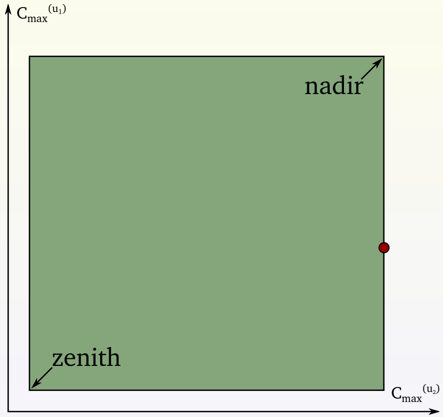
- MUSP($k : C_{\max}$) on one processor: the Earlist Deadline First algorithm gives a $<\overline{1}, \overline{1}, \ldots, \overline{1}>$-approximation .
- MUSP($1 : \sum C_i;\ k - 1 : C_{\max}$) on one processor: the Latest Starting Time (LST) algorithm gives a $<1, \overline{1}, \ldots, \overline{1}>$-approximation .
- MUSP($k : C_{\max}$) on N processors: the EDF algorithm gives a $<\overline{2}, \overline{2}, \ldots, \overline{2}>$-approximation .

Application to the MUSP problem

In the Multi-Users Scheduling Problem, denoted by MUSP($k' : \sum C_i;\ k'' : C_{\max}$), k' users are interested in minimizing the average completion time of their jobs and k'' users interested in their $C_{\max}$.

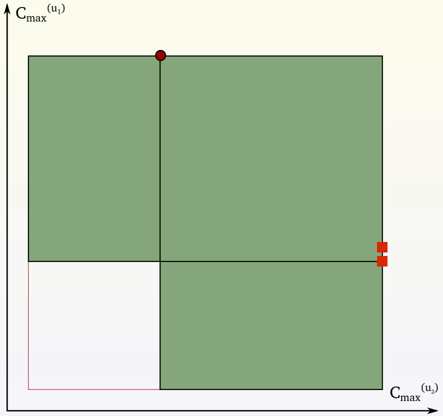
- MUSP($k : C_{\max}$) on one processor: the Earlist Deadline First algorithm gives a <$\overline{1}, \overline{1}, \ldots, \overline{1}$>-approximation .
- MUSP($1 : \sum C_i;\ k-1 : C_{\max}$) on one processor: the Latest Starting Time (LST) algorithm gives a <$1, \overline{1}, \ldots, \overline{1}$>-approximation .
- MUSP($k : C_{\max}$) on N processors: the EDF algorithm gives a <$\overline{2}, \overline{2}, \ldots, \overline{2}$>-approximation .

Decomposition of the search space



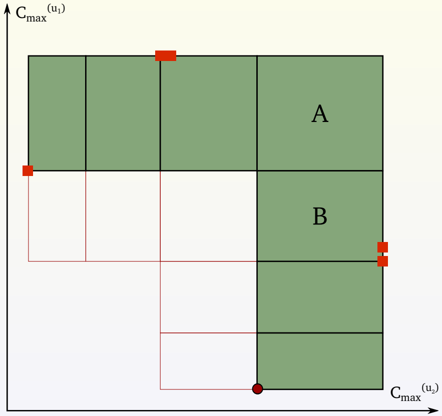
Objective space

Decomposition of the search space



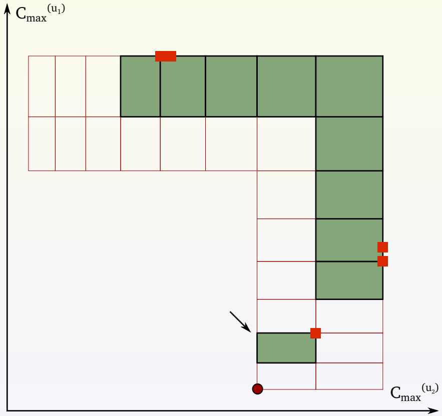
First iteration

Decomposition of the search space



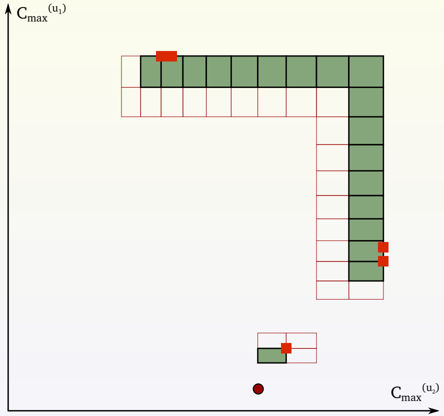
Second iteration

Decomposition of the search space



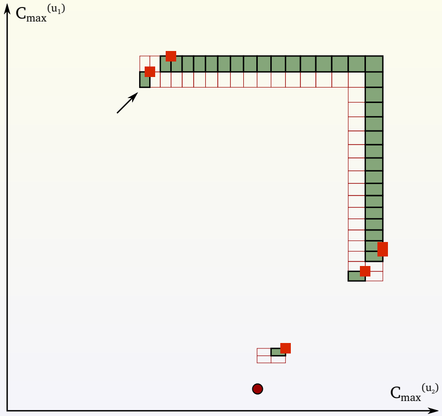
Third iteration

Decomposition of the search space



Forth iteration

Decomposition of the search space



Fifth iteration